ABSTRACT

Title of thesis: FEEDBACK CONTROL OF A HOVERCRAFT OVER A WIRELESS LINK

Zachary Kulis, Master of Science, 2006

Dissertation directed by: Professor P.S. Krishnaprasad
Department of Electrical and Computer Engineering
Dr. Eric W. Justh
Institute for Systems Research

Nonlinear underactuated systems (i.e. systems with fewer control inputs than configuration variables) present significant challenges for automatic control. This thesis explores feedback control of an underactuated hovercraft over a wireless communication channel using techniques from nonlinear control theory. A family of control laws stabilizing the hovercraft *reduced dynamics*–including zero velocity, constant forward/reverse velocity, and constant angular velocity stabilization–are derived. Lyapunov arguments are used to prove convergence of the reduced dynamics under the control laws. It is shown that heading cannot be stabilized by a continuously differentiable state feedback law. In response, two hybrid control algorithms for heading stabilization are proposed. The control laws are demonstrated on a real R/C hovercraft using a distributed autopilot and a Bluetooth network. A two-dimensional aided INS is developed using a MEMs IMU and the "Cricket" RF/ultrasonic ranging system. Experimental and simulated results from a high-fidelity model are shown to agree nicely.

FEEDBACK CONTROL OF A HOVERCRAFT OVER A WIRELESS LINK

by

Zachary Kulis

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2006

Advisory Committee:
Professor P.S. Krishnaprasad, Chair/Advisor
Dr. Eric Justh, Co-Advisor
Professor Steven Marcus

DEDICATION

Dedicated to Kirsten –

my partner in life's wonderful adventure.

ACKNOWLEDGMENTS

This thesis marks a major milestone in my study of control theory at the University of Maryland. Over the past three years, I was privileged to receive the bulk of my graduate education in control theory, including nonlinear, stochastic, optimal, and adaptive control, from a great teacher and master of the discipline, Dr. P. S. Krishnaprasad. It never ceases to amaze me that the most seemingly abstract mathematical theorems and results, when explained properly and carefully, can be readily understood and applied to solve real control problems. Dr. Krishnaprasad is a truly gifted ambassador of mathematics and a tireless champion of science.

I am equally grateful to Dr. Eric Justh, who spent many hours with me studying the nonlinear hovercraft dynamics and discussing possible strategies for control. Dr. Justh was also instrumental in providing me with valuable experimental insights, especially with regard to the INS. My research certainly would have suffered without his careful guidance and advice.

Thanks also to Dr. Steven Marcus who taught my first graduate course in control theory and could not have presented the comprehensive subject of linear control theory any more clearly.

I would also like to thank Sandy Klemm for many thoughtful discussions regarding nonlinear control of the hovercraft as well as his assistance in modifying the Cricket code for better positioning performance.

Any experimental undertaking is accompanied by its share of technical perils and practical challenges. I am especially grateful for all the assistance I received in taming these beasts, and would like to express my gratitude to Jay Renner, Shyam Mehrotra, and Jay in the machine shop who worked miracles fitting a miniature optical encoder to a Speed 400 motor shaft. I would like to thank Kevin Jackson at hovercraftmodels.com, who provided me with a fully assembled HoverDemon and answered all of my questions about potential design modifications to increase thrust production and steerability. In addition, I wish to express many thanks to Pamela White in the ISR, who was able to match my furious demand for hovercraft parts with an equally rapid placement of purchase orders.

Finally, I am especially grateful for my wonderful family, whose unwavering support is a beacon of light throughout each journey I undertake. This particular journey was made even more special by the love and support of my wonderful fiancée, Kirsten. I am extremely thankful for her patience, sacrifice, and encouragement, especially during the times when I thought that I would never finish! And of course, a big thank you goes out to Dr. Yoda, the best hovercraft pilot in the world!

This research was supported in part by the Naval Research Laboratory under Grant No. N00173-04-1G014, and by the Army Research Office under ODDR&E MURI01 Program Grant No. DAAD19-01-1-0465 to the Center for Communicating Networked Control Systems (through Boston University).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The hovercraft is a fascinating ground vehicle that possesses the unique ability to float above land or water. Riding on a cushion of air endows the hovercraft with many interesting and useful properties. Unlike wheeled robots which feature constrained kinematics, the hovercraft can move freely in any direction. For example, although the lateral direction of travel is not usually actuated, the hovercraft is completely free to move sideways. In addition, the frictional damping force acting on a hovercraft is minimal. For autonomous hovercraft applications, the lack of friction places an additional burden on the controller, as all velocity damping forces must be created by the actuators. The combination of the rich hovercraft dynamics and minimal frictional damping make the automatic control of a hovercraft a complex and interesting problem.

The concept of an *air cushion vehicle* was originally proposed in 1716 by Swedish designer, Emanuel Swedenborg. It was not until 1956, however, that the modern hovercraft was invented by British inventor Christopher Cockerell. In fact, it was Cockerell who coined the term *hovercraft* to describe his invention. The first practical hovercraft was the SR-N1, developed by Saunders Roe, a British aircraft manufacturer. Two years later, in 1961, the Vickers VA-3 became the first

commercially operated hovercraft and carried passengers regularly along the North Wales Coast.

The hovercraft achieves lift by creating a volume of high pressure air underneath the vehicle. A skirt made from flexible material encircles the underbody of the vehicle and prevents the high pressure air from rapidly escaping the *plenum* when the hovercraft lifts above the ground. A properly designed skirt is crucial to vehicle stability and ride comfort. The skirt must be flexible so that it conforms to uneven terrain, durable enough to prevent abrasion and tearing, and lightweight. For these reasons, the skirt is the most critical aspect of the entire hovercraft design [1].

A hovercraft achieves lift and propulsion by one or more high-power fans. Steering and vehicle control may be achieved in various ways. On some hovercraft, the fans may be swiveled a full 360° to produce thrust in any direction. Using two fans inline with the vehicle center of mass allows a hovercraft to turn in place simply by running the fans in opposite directions. An alternative actuation approach is to use a fixed fan and a rudder to steer. The thrust fan is usually shrouded by a duct for greater thrust efficiency, and the rudder is located in the high velocity air stream.

A rudder steering mechanism adds considerable complexity to the vehicle control. First, the fan must produce sufficient air speed for the rudder to be effective as a control surface. Thus, any turning maneuver is always accompanied by a forward (or reverse) thrust component. As a result, a hovercraft with a rudder cannot generate a pure torque and is thus unable to turn in place. Second, the rudder is mechanically limited and cannot produce force directed along the hovercraft's lat-

eral direction. For these reasons, a hovercraft with a rudder is an *underactuated system* and an interesting problem for automatic control.

Other examples of underactuated systems include robot manipulators, spacecraft, aircraft, missiles, and underwater vehicles. Despite increased control difficulty, underactuated systems can provide a substantial savings in actuator cost, size, and weight, and may be the only viable option for many applications. The control and stabilization of underactuated vehicles, however, can be quite challenging. Difficulties arise because classical nonlinear control techniques, such as feedback linearization, are not always applicable for underactuated systems [2]. Other traditional methods such as linearization and gain scheduling are popular due to their simplicity, but guarantee stability for only a local neighborhood of the operating point. Moreover, a linear controller will often perform poorly whenever the nonlinear modes of the system are exercised.

The challenges related to stabilizing an underactuated hovercraft have been partially addressed in the literature. Pettersen and Egeland [3] extended the results of Byrnes and Isidori [4] and showed that underactuated vehicles failing to satisfy a more general gravitational/bouyant field requirement cannot be stabilized by either continuous or discontinuous state feedback. In addition, the authors gave controllability results for an underactuated surface vessel (with dynamics similar to a hovercraft) and proposed a time-varying control law to stabilize the equilibrium at the origin for the dynamics.

Fantoni, Lozano, Mazenc, and Pettersen [2] addressed the problem of stabilizing the velocity and position of a disc-shaped hovercraft. The particular hovercraft

model analyzed featured dual fans offset from the center of mass. A velocity stabilization control law was shown to be globally asymptotically stable. Furthermore, the authors proposed three control laws for position stabilization (neglecting yaw angle) using the longitudinal and angular velocities as controls. One of the control laws was shown to be globally exponentially stable with control inputs that converged to zero.

Trajectory tracking for underactuated vehicles is an active area of research. A traditional approach to trajectory tracking uses the linearization of the system dynamics about a nominal state space trajectory. The problem is that the formulation of feasible state space trajectories can be particularly difficult for vehicles with complex dynamics. For example, we have shown that certain circular trajectories with rigid heading constraints are not physically realizable by our hovercraft's dynamics.

Aguiar, Cremean, and Hespanha [5] recently demonstrated an algorithm allowing a general class of underactuated vehicles to track an arbitrary reference trajectory. The algorithm is based on an iterative Lyapunov technique and yields global convergence to an arbitrarily small neighborhood of the origin. Experimental trajectory tracking results for a hovercraft-like[1] vehicle are provided.

Seguchi and Ohtsuka [6] implemented a real-time nonlinear receding horizon control algorithm for position tracking of a small R/C hovercraft. Experimental and simulated results were compared for the tracking algorithm. Finally, in [7],

[1]The vehicle features dual thrust fans and rolls on omnidirectional ball casters. Thus, significant friction is present in the dynamics.

Pettersen and Nijmeijer proposed a semi-global exponentially stable position and heading tracking control law for a surface vessel.

In this thesis, we explore nonlinear control of a hovercraft over a Bluetooth wireless link. Although the literature abounds with simulated results, our goal is to provide concrete experimental verification of the nonlinear control theory on an actual hovercraft. To make matters more challenging, we implement a real-time distributed controller with non-negligible sensing and actuation latencies, and show experimentally that the control system is robust to network delays. We also develop a two-dimensional aided inertial navigation system (INS) for measuring the hovercraft velocity and implement the system on real hardware. Unlike [5], our R/C hovercraft lifts above the ground and experiences minimal contact friction with the surface. In addition, it features a rudder for steering, making automatic control efforts more challenging than for the dual-fan hovercraft.

Starting in Chapter 2, we derive the nonlinear dynamical model of the hovercraft from first principles. We then show how the full dynamics model may be reduced to three equations for preliminary control efforts. These three equations constitute the hovercraft *reduced dynamics*. Following this, we derive a family of control laws to stabilize the reduced dynamics. We start with velocity stabilization, encountering the same result as [2], and proceed to prove convergence results for constant forward/reverse velocity stabilization and constant angular velocity stabilization. We then provide a brief discussion of the difficulties involved in stabilizing the full dynamics (six equations), citing a result proved by Byrnes and Isidori [4] and a necessary condition for stabilizability proved by Brockett [8]. We conclude the

chapter with a presentation of two hybrid control strategies for joint stabilization of the velocity and heading.

In Chapter 3, we discuss the theory of inertial navigation and explain how additional information from aiding sensors may be used to substantially increase the accuracy of an INS. We highlight the difficulties inherent in a full three-dimensional INS systems and provide a set of design assumptions for a simplified two-dimensional system. Next, we briefly explain Kalman Filtering and show how the Kalman Filter may be used to augment the performance of an INS when additional aiding measurements (such as position or heading estimates) are available. We derive the error dynamics model for a two-dimensional INS and present a simplified sensor error model for the accelerometers and gyros that accounts for the primary sources of error.

Chapter 4 marks the transition from the theoretical section of this thesis to the applied section. We begin with a description of our R/C hovercraft and discuss the construction and actuation. Next, we carefully describe the procedures followed for calibrating the hovercraft actuators. To conclude the chapter, we explain how the hovercraft moment of inertia may be determined experimentally using the INS, and provide experimental results.

In Chapter 5, we provide a full account of the hovercraft autopilot. We begin with a block diagram of the system architecture and proceed to discuss each of the subsystems in detail. We describe the dSPACE real-time processing system, which constitutes the autopilot controller, and provide a screenshot of our Simulink autopilot model. Next, we highlight the custom microcontroller solution developed to

provide low-level actuation, sensing, and communication functionality on the hovercraft. Following that, we present an overview of Bluetooth wireless technology and discuss the difficulties involved with using wireless communications in distributed control systems. To quantify the performance of our Bluetooth network, we devise an experiment to measure the uplink and downlink delay and provide experimental results. Next, we describe the INS hardware, including the IMU and aiding sensors (magnetometer and "Cricket" system), and provide the full discretized dynamics model implemented in software. To conclude the chapter, we discuss in detail the Cricket RF/ultrasonic ranging system used to aid the INS. Essentially, Cricket is an indoor GPS-replacement technology that provides position estimates with respect to a user-defined coordinate system. We document our significant modifications to the original Cricket software (developed by MIT), which yielded greatly increased positioning accuracy, and identify the remaining sources of error.

Chapter 6 provides simulated and experimental results for the hovercraft autopilot. We begin the chapter with a description of a high-fidelity nonlinear simulation developed in Simulink to capture the complete system dynamics as accurately as possible. We present simulated results for the "ideal" and "real" autopilot systems and overlay plots for direct comparison. Next, we provide experimental performance results for the aided INS system, obtained using an optical encoder and rotating platform under computer control. We compare results for optical encoder/magnetometer aiding with result for Cricket-only aiding. Finally, we present experimental results for each of the control laws derived in Chapter 1. We overlay

the experimental results with the simulated "real" autopilot plots to highlight the effectiveness of the simulator.

Finally, Chapter 7 concludes this thesis with a summary of results and directions for future research. Our hope is that this thesis and experimental work will provide a strong foundation for future research in nonlinear control theory applied to hovercraft. We elucidate several of our ambitions pertaining to the realization of a fully autonomous hovercraft in Chapter 7.

Part I

Theoretical Framework

Chapter 2

Hovercraft Control Laws

2.1 Derivation of the Hovercraft Model

Consider a right-handed inertial reference frame, U, defined by the axes (x, y, z). Assume that the z axis points into the plane, so that positive angles are measured clockwise about z. In the following analysis, we will constrain the hovercraft dynamics to the xy plane.

Let us also define a body-fixed frame, B, with axes (X, Y, Z) that is rigidly affixed to the hovercraft body, $\mathcal{B}$, at the center of mass. Assume that the X axis points from the hovercraft's tail to its nose and that the positive Z axis points into the plane.

The configuration, i.e. position and orientation, of the body in the inertial reference frame U is determined completely by a vector $\mathbf{r}$ and an angle θ. Let $\mathbf{r}$ denote the position of the body frame with respect to the inertial frame, and let θ be the angle between the x and X axes, measured from the inertial frame. Observe that the configuration space is $SE(2)$, the Special Euclidean Group of 3×3 matrices, where

$$SE(2) \triangleq \left\{ \begin{bmatrix} B & \mathbf{r} \\ 0 & 1 \end{bmatrix} \;\middle|\; B \in SO(2), \mathbf{r} \in \mathbb{R}^2 \right\} \tag{2.1}$$

In the above definition, B is the orthonormal 2×2 rotation matrix given by

$$B(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{2.2}$$

More generally, the columns of B correspond to the body frame principal axes resolved in inertial frame coordinates.

The rotation matrix $B(\theta)$ is used to transform vector quantities between these two frames. For example, a vector quantity, $\mathbf{Q}$, measured in the body frame is expressed in the inertial frame as $\mathbf{q} = B\,\mathbf{Q}$. The orthonormal property of B permits the resolution of inertial frame vector quantities in body frame coordinates by pre-multiplying the vector by B^{T}.

If we define $\Omega \triangleq \dot{\theta}$, the angular velocity of the body frame, then it may be shown that $B(\theta)$ satisfies

$$\dot{B}(\theta) = B(\theta)\,\hat{\Omega}, \tag{2.3}$$

where $\hat{\Omega}$ is the skew-symmetric matrix given by

$$\hat{\Omega} = \begin{bmatrix} 0 & -\Omega \\ \Omega & 0 \end{bmatrix}. \tag{2.4}$$

Using the notation defined above, the hovercraft kinematics are given by the following set of equations:

$$\begin{aligned} \dot{B}(\theta) &= B(\theta)\,\hat{\Omega} \\ \dot{\mathbf{r}} &= B\,\mathbf{V}, \end{aligned} \tag{2.5}$$

where $\mathbf{V}$ denotes the hovercraft velocity vector in the body-fixed frame.

We will now derive the hovercraft dynamics through a simple application of the Newton-Euler Balance Laws. In the following derivation, let $\mathbf{P}$ and $\mathbf{p}$ be the hovercraft linear momentum in body and inertial frame coordinates respectively. Additionally, $\mathbf{F}$ is the force applied by the thruster in body coordinates, $\mathbf{d}$ is the vector from the hovercraft center of mass to the rudder pivot, J denotes the moment of inertia, and m is the hovercraft mass. A graphical representation of the hovercraft model appears in Figure 2.1.

Starting with Newton's Second Law of Motion,

$$\begin{aligned}
\mathbf{p} &= m\dot{\mathbf{r}} \\
\mathbf{P} &= mB^{\mathrm{T}}\dot{\mathbf{r}} \\
\dot{\mathbf{P}} &= m\dot{B}^{\mathrm{T}}\dot{\mathbf{r}} + mB^{\mathrm{T}}\ddot{\mathbf{r}} \\
&= m\hat{\Omega}^{\mathrm{T}}B^{\mathrm{T}}\dot{\mathbf{r}} + mB^{\mathrm{T}}\ddot{\mathbf{r}} \\
\dot{\mathbf{P}} &= -\hat{\Omega}\mathbf{P} + \mathbf{F} \qquad (2.6)
\end{aligned}$$

Similarly, Euler's Balance Law relates body angular momentum $\mathbf{\Pi}$ to torque $\boldsymbol{\tau}$ by $\dot{\mathbf{\Pi}} = \boldsymbol{\tau}$. Let ϕ denote the rudder angle measured clockwise with respect to the X axis. Ideally, all of the force $\mathbf{F}$ produced by the thrust fan is directed along the angle ϕ. Applying Euler's Balance Law yields:

$$\dot{\mathbf{\Pi}} = \mathbf{d} \times \mathbf{F}. \qquad (2.7)$$

Using the formula

$$\|\dot{\mathbf{\Pi}}\| = \|\mathbf{d}\|\,\|\mathbf{F}\|\,|\sin(\phi)|, \qquad (2.8)$$

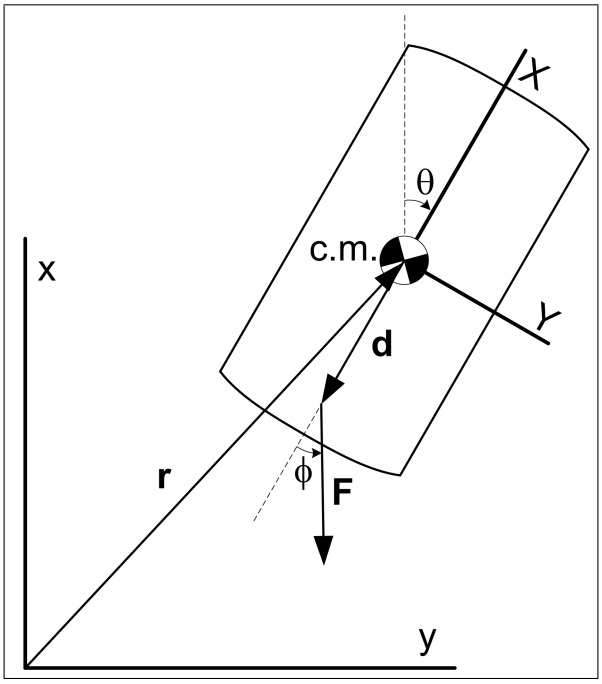


Figure 2.1: Hovercraft model used to derive the vehicle dynamics.

and the fact that rotation is confined to the plane, we may write

$$\dot{\Pi} = -\|\mathbf{d}\|\, F_Y. \tag{2.9}$$

Combining equations (2.6) and (2.9), the hovercraft dynamics are given in component form by

$$\begin{aligned} \dot{P}_X &= P_Y \frac{\Pi}{J} + F_X \\ \dot{P}_Y &= -P_X \frac{\Pi}{J} + F_Y \\ \dot{\Pi} &= -d\, F_Y \end{aligned} \tag{2.10}$$

where $d = \|\mathbf{d}\|$ and $\Pi = J\,\Omega$. In subsequent sections, we will refer to equation (2.10) as the *reduced dynamics* and equations (2.5) and (2.10) together as the *full dynamics* model.

2.2 Control Laws to Stabilize the Reduced Dynamics

As described in section 2.1, the hovercraft *full dynamics* model consists of the six equations

$$\begin{aligned}
\dot{P}_X &= P_Y \frac{\Pi}{J} + F_X \\
\dot{P}_Y &= -P_X \frac{\Pi}{J} + F_Y \\
\dot{\Pi} &= -d\, F_Y \\
\dot{\mathbf{r}} &= B(\theta) \frac{\mathbf{P}}{m} \\
\dot{B}(\theta) &= B(\theta)\, \hat{\Omega}
\end{aligned} \tag{2.11}$$

If we examine these six equations carefully, we see that the first three equations are independent of the latter three. This triangularity property allows us to analyze the first three equations separately. Recall that these equations constitute the reduced dynamics model.

We will first consider a family of control laws to stabilize the reduced dynamics. Later, we will consider the full dynamics model and seek control laws to drive the six state variables to desired values.

To begin the analysis, observe that (2.10) has the following three families of equilibria of the unforced system:

$$\begin{aligned}
&(1) \quad (P_X, P_Y, \Pi) = (0, 0, 0) \\
&(2) \quad (P_X, P_Y, \Pi) = (c_1, c_2, 0), \quad c_1, c_2 \in \mathbb{R} \\
&(3) \quad (P_X, P_Y, \Pi) = (0, 0, c), \qquad c \in \mathbb{R}
\end{aligned} \tag{2.12}$$

In contrast, the full dynamics (2.11) has a continuum of equilibria located at

$$(P_X, P_Y, \Pi, r_x, r_y, \theta) = (0, 0, 0, c_1, c_2, c_3), \quad c_1, c_2, c_3 \in \mathbb{R} \tag{2.13}$$

2.2.1 Zero Velocity Stabilization

Let us first suppose that our goal is to bring the hovercraft to rest, preferably from any arbitrary initial condition. We would like to find a control law $\mathbf{u}(t)$, such that $\lim_{t\to\infty}(P_X, P_Y, \Pi) = (0, 0, 0)$ for any $(P_X(0), P_Y(0), \Pi(0))$. In the following discussion, let $\mathbf{x} = (P_X, P_Y, \Pi)$.

Consider the Lyapunov function:

$$V(\mathbf{x}) = \frac{{P_X}^2 + {P_Y}^2}{2m} + \frac{\Pi^2}{2J} \tag{2.14}$$

This Lyapunov function represents the total energy of the system, and is the sum of the hovercraft's translational and rotational kinetic energy. We do not include a potential energy term, as the hovercraft is a planar vehicle.

A Lyapunov function of the above form is closely tied to the system dynamics. We expect that it will provide physical insight and aid in the selection of appropriate damping controls. Differentiating $V(\mathbf{x})$ with respect to t, we obtain:

$$\begin{aligned}
\dot{V}(\mathbf{x}) &= \frac{P_X}{m}\dot{P}_X + \frac{P_Y}{m}\dot{P}_Y + \frac{\Pi}{J}\dot{\Pi} \\
&= \frac{P_X}{m}\left(P_Y\frac{\Pi}{J} + F_X\right) + \frac{P_Y}{m}\left(-P_X\frac{\Pi}{J} + F_Y\right) - \frac{\Pi}{J}(d\,F_Y) \\
&= F_X\frac{P_X}{m} + F_Y\left(\frac{P_Y}{m} - d\frac{\Pi}{J}\right)
\end{aligned} \tag{2.15}$$

It is obvious from equation (2.15) that the proper choice of controls to make $\dot{V}(\mathbf{x}) \leq 0$ is:

$$\begin{cases} F_X &= -k_1 \frac{P_X}{m} \\ F_Y &= -k_2 \left(\frac{P_Y}{m} - d\frac{\Pi}{J} \right), \quad k_1, k_2 > 0 \end{cases} \tag{2.16}$$

Substituting for (F_X, F_Y) in equation (2.15) yields

$$\dot{V}(\mathbf{x}) = -k_1 \frac{{P_X}^2}{m^2} - k_2 \left(\frac{P_Y}{m} - d\,\frac{\Pi}{J} \right)^2 \leq 0. \tag{2.17}$$

The control law specified in equation (2.16) forces the first time derivative of $V(\mathbf{x})$ to be nonpositive. Through a simple application of LaSalle's Invariance Principle, we will show that the control law drives the state (P_X, P_Y, Π) asymptotically to the origin.

Theorem 2.2.1 LaSalle's Invariance Principle [9]

Let D be a domain of the reduced dynamics (2.10) *and let $\Gamma \subset D$ be a compact set that is positively invariant with respect to* (2.10). *Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that $\dot{V}(\mathbf{x}) \leq 0$ in Γ. Let E be the set of all points in Γ where $\dot{V}(x) = 0$. Let M be the largest invariant set in E. Then every solution starting in Γ approaches M as $t \to \infty$.*

Proposition 2.2.2 *Control law* (2.16) *with $d, m, J, k_1, k_2 > 0$ asymptotically stabilizes the origin of the hovercraft reduced dynamics. Moreover, the origin is globally asymptotically stable.*

Proof: Consider the set $E \triangleq \left\{ \mathbf{x} : \dot{V}(\mathbf{x}) = 0 \right\}$. Necessarily, $P_X = 0$ is in E. This implies that $F_X = 0$ in (2.16) (fact 1). Also, it is required that $\frac{P_Y}{m} - d\frac{\Pi}{J} = 0$, which

implies that $P_Y = \frac{\Pi}{J} md$ is in E (fact 2). Now, from the system dynamics (2.10) and fact 1, $\dot{P}_X = \frac{\Pi}{J} P_Y + F_X = \frac{\Pi}{J} P_Y$. Since $\dot{P}_X \equiv 0$, $\frac{\Pi}{J} P_Y = 0$. Thus, either P_Y or $\Pi = 0$. This observation, together with fact 2, implies that $P_Y = \Pi = 0$. Therefore, the origin is the only point in the set E, and consequently the only point in the set M. Since $V(\mathbf{x})$ is radially unbounded, we may chose Γ to be arbitrarily large. Thus, control law (2.16) ensures that $(P_X, P_Y, \Pi) \equiv 0$ is a globally asymptotically stable equilibrium. ■

In the next sections, we will adapt Lyapunov function (2.14) and the above analysis to stabilize an arbitrary forward or reverse momentum, $\overline{P}_X$.

2.2.2 Constant $\overline{P}_X$ Stabilization

Building on our analysis in the previous section, we will now find a control law to stabilize a constant arbitrary forward momentum, $\overline{P}_X$. Such a control law will be useful both for pilot-in-the-loop applications, where active damping of P_Y and Π is desired, and also in approximating reference trajectories with straight line segments.

We begin the analysis by defining new hat state variables. Let

$$\begin{aligned} \hat{P}_X &= P_X - \overline{P}_X \\ \hat{P}_Y &= P_Y \\ \hat{\Pi} &= \Pi \end{aligned} \tag{2.18}$$

Lyapunov function (2.14) is also modified slightly and takes the following form, with $\mathbf{x} \triangleq \left(\hat{P}_X, \hat{P}_Y, \hat{\Pi}\right)$:

$$W(\mathbf{x}) = \frac{\hat{P}_X^2 + \hat{P}_Y^2}{2m} + \frac{\alpha\,\hat{\Pi}^2}{2J}, \quad \alpha > 0 \tag{2.19}$$

The need for the scalar α will soon become apparent.

Taking time derivatives of these new state variables, and letting $F_X = -k_1 \frac{\hat{P}_X}{m}$ and $F_Y = -k_2 \left(\frac{\hat{P}_Y}{m} - d\,\frac{\hat{\Pi}}{J}\right)$ as before yields:

$$\begin{aligned}
\dot{\hat{P}}_X &= \hat{P}_Y\,\frac{\hat{\Pi}}{J} - k_1 \frac{\hat{P}_X}{m} \\
\dot{\hat{P}}_Y &= -\left(\hat{P}_X + \overline{P}_X\right)\,\frac{\hat{\Pi}}{J} - k_2 \left(\frac{\hat{P}_Y}{m} - d\,\frac{\hat{\Pi}}{J}\right) \\
\dot{\hat{\Pi}} &= k_2\,d\left(\frac{\hat{P}_Y}{m} - d\,\frac{\hat{\Pi}}{J}\right)
\end{aligned} \tag{2.20}$$

By defining a new variable, $\hat{d} \triangleq d - \frac{\overline{P}_X}{k_2}$, and performing the necessary substitutions and groupings in equation (2.20), we obtain the following dynamical equations:

$$\begin{aligned}
\dot{\hat{P}}_X &= \hat{P}_Y\,\frac{\hat{\Pi}}{J} - k_1 \frac{\hat{P}_X}{m} \\
\dot{\hat{P}}_Y &= -\hat{P}_X\,\frac{\hat{\Pi}}{J} - k_2 \left(\frac{\hat{P}_Y}{m} - \hat{d}\,\frac{\hat{\Pi}}{J}\right) \\
\dot{\hat{\Pi}} &= k_2\,d\left(\frac{\hat{P}_Y}{m} - \hat{d}\,\frac{\hat{\Pi}}{J}\right) - d\,\overline{P}_X\,\frac{\hat{\Pi}}{J}
\end{aligned} \tag{2.21}$$

We may now proceed to compute $\dot{W}(\mathbf{x})$ for $W(\mathbf{x})$ defined in (2.19). This yields:

$$\begin{aligned}
\dot{W}(\mathbf{x}) = \;& \tfrac{\hat{P}_X}{m}\left(\hat{P}_Y\,\tfrac{\hat{\Pi}}{J} - \tfrac{k_1}{m}\,\hat{P}_X\right) + \\
& \tfrac{\hat{P}_Y}{m}\left[-\hat{P}_X\,\tfrac{\hat{\Pi}}{J} - k_2\left(\tfrac{\hat{P}_Y}{m} - \hat{d}\,\tfrac{\hat{\Pi}}{J}\right)\right] + \\
& \alpha\,\tfrac{\hat{\Pi}}{J}\left[k_2\,d\left(\tfrac{\hat{P}_Y}{m} - \hat{d}\,\tfrac{\hat{\Pi}}{J}\right) - d\,\overline{P}_X\,\tfrac{\hat{\Pi}}{J}\right]
\end{aligned} \tag{2.22}$$

Factoring (2.22) and letting $\alpha = \frac{\hat{d}}{d}$ results in the following simplified equation:

$$\begin{aligned}\dot{W}(\mathbf{x}) &= -k_1 \frac{\hat{P}_X^2}{m^2} - k_2 \left(\frac{\hat{P}_Y}{m} - \hat{d}\,\frac{\hat{\Pi}}{J}\right)\left(\frac{\hat{P}_Y}{m} - \alpha\, d\,\frac{\hat{\Pi}}{J}\right) - \alpha\, d\,\overline{P}_X \frac{\hat{\Pi}^2}{J^2} \\ &= -k_1 \frac{\hat{P}_X^2}{m^2} - k_2 \left(\frac{\hat{P}_Y}{m} - \hat{d}\,\frac{\hat{\Pi}}{J}\right)^2 - \hat{d}\,\overline{P}_X \frac{\hat{\Pi}^2}{J^2} \end{aligned} \tag{2.23}$$

It is now evident from (2.23) that $\dot{W}(\mathbf{x}) < 0$, if $\overline{P}_X > 0$ and $\hat{d} > 0$. The second condition will be satisfied if we choose $k_2 > \frac{\overline{P}_X}{d}$. Therefore, since $\dot{W}(\mathbf{x})$ is negative definite, the reduced dynamics are asymptotically driven to $(\overline{P}_X, 0, 0)$ by the control law:

$$\begin{cases} F_X &= -k_1 \frac{P_X - \overline{P}_X}{m} \\ F_Y &= -k_2 \left(\frac{P_Y}{m} - d\frac{\Pi}{J}\right), \quad k_1, \overline{P}_X > 0, \;\; k_2 > \frac{\overline{P}_X}{d} \end{cases} \tag{2.24}$$

Furthermore, the equilibrium $(\overline{P}_X, 0, 0)$ is globally asymptotically stable since $W(\mathbf{x})$ is positive definite and radially unbounded. Stabilizing $\overline{P}_X < 0$ requires additional analysis and is discussed later in section 2.2.4.

2.2.3 Alternate Derivation of a Control Law Stabilizing $\overline{P}_X > 0$

Rather than merely guessing the correct form of the control law to stabilize positive $\overline{P}_X$ as we did in section 2.2.2, let us now consider all possible linear control laws of the form:

$$\begin{bmatrix} F_X \\ F_Y \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} \hat{P}_X \\ \hat{P}_Y \\ \hat{\Pi} \end{bmatrix} \tag{2.25}$$

where $\hat{P}_X = P_X - \overline{P}_X$, $\hat{P}_Y = P_Y$, and $\hat{\Pi} = \Pi$ as before.

Substituting for F_X and F_Y in the dynamics equations given by

$$\begin{aligned}
\dot{\hat{P}}_X &= \hat{P}_Y \frac{\hat{\Pi}}{J} + F_X \\
\dot{\hat{P}}_Y &= -\left(\hat{P}_X + \overline{P}_X\right) \frac{\hat{\Pi}}{J} + F_Y \\
\dot{\hat{\Pi}} &= -d\, F_Y
\end{aligned} \tag{2.26}$$

and computing $\dot{W}(\mathbf{x})$ using (2.19) with $\alpha = 1$ yields:

$$\begin{aligned}
\dot{W}(\mathbf{x}) = & a_{11} \tfrac{\hat{P}_X^2}{m} + a_{22} \tfrac{\hat{P}_Y^2}{m} - a_{23} \tfrac{d\hat{\Pi}^2}{J} \\
& + \hat{P}_X \hat{P}_Y \left(\tfrac{a_{12}+a_{21}}{m}\right) \\
& + \hat{P}_X \hat{\Pi} \left(\tfrac{a_{13}}{m} - \tfrac{a_{21}\, d}{J}\right) \\
& + \hat{P}_Y \hat{\Pi} \left(\tfrac{a_{23}}{m} - \tfrac{a_{22}\, d}{J} - \tfrac{\overline{P}_X}{mJ}\right)
\end{aligned} \tag{2.27}$$

It is clear that we can eliminate the cross terms and make $\dot{W}(\mathbf{x}) < 0$ by forcing the following conditions on the free variables:

$$\begin{aligned}
&(1) && a_{11}, a_{22} && < && 0 \\
&(2) && a_{23} && > && 0 \\
&(3) && a_{12} && = && -a_{21} \\
&(4) && \tfrac{a_{13}}{m} && = && \tfrac{a_{21}\, d}{J} \\
&(5) && \tfrac{a_{23}}{m} - \tfrac{a_{22}\, d}{J} && = && \tfrac{\overline{P}_X}{mJ}
\end{aligned}$$

We will now show that conditions (1), (2), and (5) are incompatible for $\overline{P}_X < 0$. Solving for a_{22} in condition (5) gives $a_{22} = \frac{a_{23}\, J - \overline{P}_X}{dm}$. If $\overline{P}_X < 0$ and $a_{23} > 0$ as in condition (2), then $a_{22} > 0$. This, however, contradicts condition (1). Thus, we must restrict $\overline{P}_X > 0$.

One choice of scalars that satisfies the above conditions and results in a control law with the same form as equation (2.24) is:

$$\begin{aligned} a_{12} = \quad a_{21} = a_{13} = 0 \\ a_{11} = -\tfrac{k_1}{m} \\ a_{23} = \quad \tfrac{\overline{P}_X}{2J} \\ a_{22} = -\tfrac{\overline{P}_X}{2dm} \end{aligned} \tag{2.28}$$

Substituting into (2.25) yields the following control law:

$$\begin{cases} F_X &= -k_1 \, \frac{P_X - \overline{P}_X}{m} \\ F_Y &= -\frac{\overline{P}_X}{2d} \left(\frac{P_Y}{m} - d \, \frac{\Pi}{J} \right), \quad \overline{P}_X, k_1 > 0 \end{cases} \tag{2.29}$$

In the above equation for F_Y, $\frac{\overline{P}_X}{2d}$ plays the role of k_2 in equation (2.24). Note, however, that while $k_2 > \frac{\overline{P}_X}{d}$ was required in (2.24), $\frac{\overline{P}_X}{2d} < \frac{\overline{P}_X}{d} = k_2$. Under control law (2.29), the origin of dynamics (2.26) is globally asymptotically stable, since the Lyapunov function, $W(\mathbf{x})$ in equation (2.19) with $\alpha = 1$, is radially unbounded.

2.2.4 $\overline{P}_X < 0$ Stabilization

In the previous section, we derived two control laws to stabilize the origin in $(\hat{P}_X, \hat{P}_Y, \hat{\Pi})$ space. In both derivations, we required $\overline{P}_X > 0$ to make the Lyapunov stability arguments hold. In this section, we consider the stabilization of $(\overline{P}_X, 0, 0)$ when $\overline{P}_X$ is negative. We use Lyapunov's indirect method to show that the origin is locally asymptotically stable.

Consider the following control law:

$$\begin{cases} F_X &= -\frac{k_1}{m}\left(P_X - \overline{P}_X\right) \\ F_Y &= k_2\left(\frac{P_Y}{m} + \beta d\frac{\Pi}{J}\right), \quad k_1, k_2, \beta > 0, \quad \overline{P}_X < 0 \end{cases}$$

Substituting for F_X and F_Y in the reduced dynamics (2.10) produces the following vector field, $f(\mathbf{x})$:

$$f(\mathbf{x}) = \begin{bmatrix} P_Y\frac{\Pi}{J} - \frac{k_1}{m}\left(P_X - \overline{P}_X\right) \\ -P_X\frac{\Pi}{J} + k_2\left(\frac{P_Y}{m} + \beta d\frac{\Pi}{J}\right) \\ -k_2 d\left(\frac{P_Y}{m} + \beta d\frac{\Pi}{J}\right) \end{bmatrix} \tag{2.30}$$

The Jacobian of (2.30) evaluated at $(\overline{P}_X, 0, 0)$ is:

$$\left.\frac{\partial f}{\partial x}\right|_{(\overline{P}_X,0,0)} = \begin{bmatrix} -\frac{k_1}{m} & 0 & 0 \\ 0 & \frac{k2}{m} & \frac{k_2 d\beta - \overline{P}_X}{J} \\ 0 & -\frac{k_2 d}{m} & -\frac{k_2 d^2\beta}{J} \end{bmatrix} \tag{2.31}$$

and the characteristic equation is:

$$\chi(s) = \left(s + \frac{k_1}{m}\right)\left[s^2 + k_2\left(\frac{d^2\beta}{J} - \frac{1}{m}\right)s - \frac{k_2\overline{P}_X d}{mJ}\right] \tag{2.32}$$

Solving the roots of the characteristic equation yields the following eigenvalues:

$$\begin{aligned} s_1 &= -\frac{k_1}{m} \\ s_2, s_3 &= -\frac{k_2}{2mJ}\left(d^2\beta m - J\right) \pm \frac{\sqrt{{k_2}^2(d^2\beta m - J)^2 + 4k_2\overline{P}_X dmJ}}{2mJ} \end{aligned} \tag{2.33}$$

We wish to find conditions on β such that the eigenvalues have negative real parts. First, setting $d^2\beta m - J > 0$ implies that $\beta > \frac{J}{d^2 m}$. Since $4k_2\overline{P}_X dmJ < 0$ for $\overline{P}_X < 0$, the root term in (2.33), if real-valued, must be less than $\frac{\sqrt{{k_2}^2(d^2\beta m - J)^2}}{2mJ} = \frac{k_2}{2mJ}\left|d^2\beta m - J\right|$. Therefore, $\Re\{s_2, s_3\} < 0$.

We have shown that for $\beta > \frac{J}{d^2 m}$, control law (2.30) with $\overline{P}_X < 0$ stabilizes the point $(\overline{P}_X, 0, 0)$ for initial conditions in a local neighborhood of the equilibrium. Unfortunately, we have not yet been successful in finding a radially unbounded Lyapunov function to prove global convergence using Lyapunov's direct method. Simulating the dynamics under the proposed control law, however, provides a strong indication that the equilibrium has a large region of attraction and may even be globally asymptotically stable.

2.2.5 Constant $\overline{\Pi}$ Stabilization

In this section, we consider turning the hovercraft in place at a constant angular velocity. We will derive an asymptotically stable control law that uses only the reduced dynamics. Note that our goal is not to turn the hovercraft about a prescribed *fixed* coordinate in the inertial frame. Such a control law would involve two additional states, R_X and R_Y, and be more difficult to derive. In fact, we will show later in section 2.3 that the failure of a necessary rank condition implies that the full dynamics can not be stabilized by a continuously differentiable feedback law using only the state variables.

The control law derived in this section asymptotically drives the linear momentum to zero, while achieving a constant arbitrary angular momentum, $\overline{\Pi}$. This control law, together with the zero velocity stabilization law, will later be used in a pointing algorithm to maintain a desired heading.

Consider the reduced hovercraft dynamics, (2.10). Our goal is to stabilize the point $(0, 0, \overline{\Pi})$ for an arbitrary constant $\overline{\Pi}$. To begin the analysis, define the following hat variables:

$$\begin{aligned} \hat{P}_X &= P_X \\ \hat{P}_Y &= P_Y \\ \hat{\Pi} &= \Pi - \overline{\Pi} \end{aligned} \tag{2.34}$$

Substituting into the dynamics (2.10) we get:

$$\begin{aligned} \dot{\hat{P}}_X &= \hat{P}_Y \frac{\hat{\Pi} + \overline{\Pi}}{J} + F_X \\ \dot{\hat{P}}_Y &= -\hat{P}_X \frac{\hat{\Pi} + \overline{\Pi}}{J} + F_Y \\ \dot{\hat{\Pi}} &= -d\, F_Y \end{aligned} \tag{2.35}$$

Additionally, consider the Lyapunov function candidate:

$$W(\mathbf{x}) = \frac{\hat{P}_X^2 + \hat{P}_Y^2}{2m} + \frac{\hat{\Pi}^2}{2J} \tag{2.36}$$

This Lyapunov function will lead us to a control law that drives the system to $(0, 0, \overline{\Pi})$. It will also be used to prove stability of the equilibrium.

Differentiating $W(\mathbf{x})$ with respect to time yields:

$$\begin{aligned} \dot{W}(\mathbf{x}) &= \frac{\hat{P}_X}{m}\left[\hat{P}_Y\left(\frac{\hat{\Pi} + \overline{\Pi}}{J}\right) + F_X\right] + \frac{\hat{P}_Y}{m}\left[-\hat{P}_X\left(\frac{\hat{\Pi} + \overline{\Pi}}{J}\right) + F_Y\right] - \frac{\hat{\Pi}}{J} d\, F_Y \\ &= \frac{\hat{P}_X}{m} F_X + \left(\frac{\hat{P}_Y}{m} - d\,\frac{\hat{\Pi}}{J}\right) F_Y \end{aligned} \tag{2.37}$$

Choosing $F_X = -k_1 \frac{\hat{P}_X}{m}$ and $F_Y = -k_2\left(\frac{\hat{P}_Y}{m} - d\,\frac{\hat{\Pi}}{J}\right)$, with constants $k_1, k_2 > 0$, ensures that $\dot{W}(\mathbf{x}) \leq 0$.

Unfortunately, a direct application of LaSalle's Invariance Principle can not be used to prove stability of the equilibrium $(0, 0, \overline{\Pi})$. To see why, consider the set $E \triangleq \left\{ \mathbf{x} : \dot{W}(\mathbf{x}) = 0 \right\}$. Equation (2.37) implies that two conditions must be met for all points $\mathbf{x}$ in E. They are:

$$\begin{array}{ll} (1) & \hat{P}_X = 0 \implies F_X = 0 \text{ and } \dot{\hat{P}}_X \equiv 0 \\ (2) & \frac{\hat{P}_Y}{m} - d\,\frac{\hat{\Pi}}{J} = 0 \implies \hat{P}_Y = md\frac{\hat{\Pi}}{J} \end{array} \tag{2.38}$$

From the system dynamics (2.35) and condition (1), $\dot{\hat{P}}_X \equiv 0 \implies \hat{P}_Y\,(\hat{\Pi} + \overline{\Pi}) = 0$, which further implies that either $\hat{P}_Y = 0$ or $\hat{\Pi} + \overline{\Pi} = 0$. First, consider the case with $\hat{P}_Y = 0$. Condition (2) implies that $\hat{\Pi} = 0$. Using the definition of $\hat{\Pi}$, this is equivalent to $\Pi = \overline{\Pi}$. Thus, the point $(P_X, P_Y, \Pi) = (0, 0, \overline{\Pi})$ is in the set E.

On the other hand, it may be the case that $\hat{\Pi} + \overline{\Pi} = 0$. Condition (2) then implies that $\hat{P}_Y = -md\frac{\overline{\Pi}}{J}$. Clearly, the point $(P_X, P_Y, \Pi) = (0, -md\frac{\overline{\Pi}}{J}, 0)$ is also in E.

Observe that $(0, 0, \overline{\Pi})$ and $(0, -md\frac{\overline{\Pi}}{J}, 0)$ are also in the set M, since both points are equilibria of (2.35) under the control law:

$$\left\{ \begin{array}{rcl} F_X &=& -k_1\,\frac{P_X}{m} \\ F_Y &=& -k_2\left(\frac{P_Y}{m} - d\,\frac{\Pi - \overline{\Pi}}{J}\right), \quad k_1, k_2 > 0 \end{array} \right. \tag{2.39}$$

Invoking LaSalle's Invariance Principle allows us to conclude that the control law guarantees convergence to one of these two equilibria, but it is not yet clear how to steer the system dynamics to the desired limit point.

2.2.6 Resolution of the Limit Point Ambiguity when Stabilizing $\overline{\Pi}$

Given that control law (2.39) drives the system to either $(0, 0, \overline{\Pi})$ or $(0, -md\frac{\overline{\Pi}}{J}, 0)$, one might ask if there is a set of conditions that guarantees convergence to a particular equilibrium. Specifically, we would like to derive a set of conditions that will force the dynamics to the desired equilibrium, $(0, 0, \overline{\Pi})$.

Consider Lyapunov function (2.36) with the hat variables defined as in the previous section. Plugging in the first equilibrium makes $W(\mathbf{x}) = 0$. Similarly, the *bad* equilibrium yields $W(\mathbf{x}) = \frac{\left(dm\overline{\Pi}\right)^2}{2mJ^2} + \frac{\overline{\Pi}^2}{2J}$. We will denote this value as W_{bad}. Thus, $(0, 0, \overline{\Pi})$ is a global minimizer of $W(\mathbf{x})$, while $(0, -md\frac{\overline{\Pi}}{J}, 0)$ is only a local minimizer. Since $\dot{W}(\mathbf{x}) \leq 0$, any initial condition, $\mathbf{x_0}$, chosen such that $W(\mathbf{x_0}) < W_{\text{bad}}$ will cause the system to be driven to the correct equilibrium.

We will now illustrate the relationship between a desired $\overline{\Pi}$ and the initial conditions sufficient to ensure that control law (2.39) drives the system to $(0, 0, \overline{\Pi})$. To begin, assume that control law (2.16) has been used to drive the hovercraft sufficiently to rest. By sufficiently, we mean that the Lyapunov function

$$V(\mathbf{x}) = \frac{{P_X}^2 + {P_Y}^2}{2m} + \frac{\Pi^2}{2J} \tag{2.40}$$

has attained a small positive value, ε. Note that $V(\mathbf{x}) < \varepsilon$ implies that $\frac{\Pi^2}{2J} < \varepsilon$, which further implies that $|\Pi| < \sqrt{2\varepsilon J}$. Replacing the hat variables with their definitions in equation (2.36) gives:

$$\begin{aligned} W(\mathbf{x}) &= \frac{{P_X}^2 + {P_Y}^2}{2m} + \frac{\left(\Pi - \overline{\Pi}\right)^2}{2J} \\ &= V(\mathbf{x}) - \frac{\Pi\,\overline{\Pi}}{J} + \frac{\overline{\Pi}^2}{2J} \\ &< \varepsilon + \left|\frac{\Pi\,\overline{\Pi}}{J}\right| + \frac{\overline{\Pi}^2}{2J} \end{aligned} \tag{2.41}$$

Now, setting $W(\mathbf{x_0}) < W_{\text{bad}}$ is sufficient to ensure convergence to the proper equilibrium. The analysis proceeds as follows:

$$\begin{aligned} \varepsilon + \left|\frac{\Pi\,\overline{\Pi}}{J}\right| + \frac{\overline{\Pi}^2}{2J} &< \frac{\left(dm\overline{\Pi}\right)^2}{2mJ^2} + \frac{\overline{\Pi}^2}{2J} \\ \varepsilon + \left|\frac{\Pi\,\overline{\Pi}}{J}\right| &< \frac{d^2m\left|\overline{\Pi}\right|^2}{2J^2} \end{aligned} \tag{2.42}$$

Recall that $|\Pi| < \sqrt{2\varepsilon J}$ and observe that the l.h.s. of equation (2.42) is bounded above by $\varepsilon + \frac{|\overline{\Pi}|}{J}\,|\Pi|$. Thus, we may proceed for the worst case by substituting the l.h.s. with the upper bound:

$$\begin{aligned} \varepsilon + \frac{|\overline{\Pi}|}{J}\sqrt{2\varepsilon J} &< \frac{d^2m\left|\overline{\Pi}\right|^2}{2J^2} \\ \varepsilon J^2 + J\sqrt{2\varepsilon J}\,\left|\overline{\Pi}\right| - \frac{d^2m\left|\overline{\Pi}\right|^2}{2} &< 0 \end{aligned} \tag{2.43}$$

Rewriting (2.43), the resulting sufficient condition is:

$$\frac{d^2m\left|\overline{\Pi}\right|^2}{2} - J\sqrt{2\varepsilon J}\,\left|\overline{\Pi}\right| - \varepsilon J^2 > 0 \tag{2.44}$$

We will now find conditions on $\left|\overline{\Pi}\right|$ and ε such that condition (2.44) is satisfied. Define $f(v) = \frac{d^2 m|v|^2}{2} - J\sqrt{2\varepsilon J}\ |v| - \varepsilon J^2$. Differentiating with respect to v and setting the resulting expression equal to 0 yields:

$$\begin{aligned} f'(v) &= (d^2 m)\, v - J\sqrt{2\varepsilon J}, \quad v > 0 \\ f'(v) &= (d^2 m)\, v + J\sqrt{2\varepsilon J}, \quad v < 0 \\ |v| &= \frac{J\sqrt{2\varepsilon J}}{d^2 m} \end{aligned} \tag{2.45}$$

We evaluate $f(v)$ at either critical point and obtain:

$$\begin{aligned} f(v)\big|_{v=\pm\frac{J\sqrt{2\varepsilon J}}{d^2 m}} &= \frac{d^2 m}{2}\left(\frac{2\varepsilon J^3}{d^4 m^2}\right) - J\sqrt{2\varepsilon J}\left(\frac{J\sqrt{2\varepsilon J}}{d^2 m}\right) - \varepsilon J^2 \\ &= \frac{\varepsilon J^3}{d^2 m} - \frac{2\varepsilon J^3}{d^2 m} - \varepsilon J^2 \\ &= -\varepsilon J^2\left(\frac{J}{d^2 m} + 1\right) < 0 \end{aligned} \tag{2.46}$$

Computing $f''(v) = d^2 m > 0$ shows that both of these critical points are minima. This result guarantees that a $\overline{\Pi}$ exists satisfying condition (2.44). To find the valid $\overline{\Pi}$ values, we first need to solve $f(\overline{\Pi}) = 0$. We will denote the solutions by ${\overline{\Pi}_+}^\star$ and ${\overline{\Pi}_-}^\star$, for $\overline{\Pi} > 0$ and $\overline{\Pi} < 0$, respectively.

$$\begin{aligned} {\overline{\Pi}_+}^\star &= \frac{\sqrt{2\varepsilon J^3} + \sqrt{2\varepsilon\left(J^3 + d^2 m J^2\right)}}{d^2 m} \\ {\overline{\Pi}_-}^\star &= -\left(\frac{\sqrt{2\varepsilon J^3} + \sqrt{2\varepsilon\left(J^3 + d^2 m J^2\right)}}{d^2 m}\right) \end{aligned} \tag{2.47}$$

Referring to condition (2.44), we need only to choose $\overline{\Pi} > {\overline{\Pi}_+}^\star > 0$ or $\overline{\Pi} < {\overline{\Pi}_-}^\star < 0$. This is accomplished by letting $\left|\overline{\Pi}\right| = \frac{2\sqrt{2\varepsilon(J^3 + d^2 m J^2)}}{d^2 m}$. Solving for ε, we obtain:

$$\varepsilon(\overline{\Pi}) = \frac{\left(d^2 m\right)^2 \overline{\Pi}^2}{8\, J^2\left(J + d^2 m\right)} \tag{2.48}$$

Equation (2.48) prescribes the degree to which the hovercraft should be brought to rest before switching to the $\overline{\Pi}$ stabilization control law. We see from the quadratic dependence on $\overline{\Pi}$ that the tolerance becomes less restrictive as the target rate of rotation is increased. Additionally, achieving a low rate of rotation imposes a larger burden on the controller. For small $\overline{\Pi}$, we must wait longer for the tolerance, ε, to be met before switching to the $\overline{\Pi}$ stabilization law. Thus, (2.48) has important implications for practical implementations of the control law, since a lower bound may exist on the achievable ε due to noise and other system disturbances.

2.3 Control Laws to Stabilize the Full Dynamics

So far, we have analyzed a family of control laws to stabilize the reduced dynamics. We derived and proved convergence properties for control laws that stabilize the origin in (P_X, P_Y, Π) space, stabilize constant $\overline{P}_X$, and stabilize constant $\overline{\Pi}$. In this section, we will investigate the problem of stabilizing the full dynamics in $(P_X, P_Y, \Pi, r_x, r_y, \theta)$ six-dimensional space.

2.3.1 Stabilization of the Origin

At this point, it is not even clear whether a continuously differentiable state feedback control law exists to stabilize the full dynamics (2.11). Should such a law exist, we expect it will be difficult to find, given that the hovercraft is not actuated in the lateral direction. To answer the existence question, we turn to a

powerful theorem of Byrnes and Isidori [4], which provides a necessary and sufficient condition.

Theorem 2.3.1 (Byrnes and Isidori Theorem)

Consider dynamics of the form:

$$\begin{aligned} \dot{x}_2 &= f_2(x_1, x_2), \quad x_2 \in \mathbb{R}^{n_2} \\ \dot{x}_1 &= f_1(x_1, x_2)\, x_1 + \sum_{i=1}^{m} b_i u_i, \quad u_i \in \mathbb{R},\ \ x_1, b_i \in \mathbb{R}^{n_1} \end{aligned} \tag{2.49}$$

Assume that

$$f(x) = \begin{bmatrix} f_1(x)x_1 \\ \\ f_2(x) \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ \\ x_2 \end{bmatrix} \in \mathbb{R}^{n_1+n_2} \tag{2.50}$$

is in C^∞ and has an equilibrium point at 0. Also, assume that the following conditions are satisfied:

$$\begin{aligned} H1: &\quad f_2(x) = 0 \implies x_1 = 0 \\ H2: &\quad \tfrac{\partial f_2}{\partial x_1}(0) \text{ has rank } n_2 \end{aligned} \tag{2.51}$$

Let $m' = \dim\left(\text{span}\{b_1, \ldots, b_m\}\right)$. Then, there is a continuously differentiable feedback law, $u_i = F_i(x)$, rendering the origin locally asymptotically stable $\iff$ $m' = n_1$.

Proposition 2.3.2 [10] *The full hovercraft dynamics* (2.11) *cannot be stabilized by a continuously differentiable feedback control law of the form $u = F(\mathbf{x})$, where $u = [F_X, F_Y]^{\mathrm{T}}$, $F(\mathbf{x}) = [F_1(\mathbf{x}), F_2(\mathbf{x})]^{\mathrm{T}}$, and $\mathbf{x} = [P_X, P_Y, \Pi, r_x, r_y, \theta]^{\mathrm{T}}$.*

Proof: The full hovercraft dynamics can be cast into the desired form by letting

$$x_1 = \begin{bmatrix} P_X \\ P_Y \\ \Pi \end{bmatrix}, \qquad x_2 = \begin{bmatrix} \theta \\ R_X \\ R_Y \end{bmatrix}$$

$$f_1(x) = \begin{bmatrix} 0 & \frac{\Pi}{J} & 0 \\ -\frac{\Pi}{J} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad f_2(x) = \begin{bmatrix} \frac{\Pi}{J} \\ \frac{P_X}{m} \\ \frac{P_Y}{m} \end{bmatrix}$$

$$b_1(x) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad b_2(x) = \begin{bmatrix} 0 \\ 1 \\ -d \end{bmatrix}$$

where $n_1 = n_2 = 3$.

We begin by checking condition $H1$. $f_2(x) = 0$ implies that $(\Pi, P_X, P_Y) = 0$, which is equivalent to $x_1 = 0$. Thus, condition $H1$ is satisfied. To check condition $H2$, we first compute

$$\frac{\partial f_2}{\partial x_1} = \begin{bmatrix} 0 & 0 & \frac{1}{J} \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \end{bmatrix} \tag{2.52}$$

and then note that $\operatorname{rank}\left(\frac{\partial f_2}{\partial x_1}\right) = 3 = n_2$. Condition $H2$ is also satisfied. We now compute the span of the input vector field and obtain:

$$m' = \dim\left(\operatorname{span}\left\{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -d \end{bmatrix}\right\}\right) = 2 \tag{2.53}$$

Hence, $m' \neq n_1$. Therefore, citing Theorem 2.3.1, we conclude that a continuously differentiable stabilizing feedback law does not exist. ■

While Theorem 2.3.1 precludes the existence of a certain class of feedback laws, all is not lost. We expect that a time-varying control law of the form $u = F(t, x)$ or a hybrid control scheme might achieve the desired goal. A hybrid controller consists of n distinct control laws

$$f_i = u_i(t, x), \quad i = 1, 2, \ldots, n \tag{2.54}$$

and a logic function

$$g : (t, x, v) \longmapsto f_i(t, x), \quad x \in \mathbb{R}^n, v \in \mathbb{R}^q \tag{2.55}$$

that switches between the control laws based on the system state x and additional system status v. For example, an algorithm for stabilizing the origin in six-dimensional space might consist of the following steps:

(1) Compute the position error vector given the current location.

(2) Turn in place to align the hovercraft with the error vector.

(3) Move forward (or backward) until a prescribed positioning tolerance is met. The reference velocity should decrease as the hovercraft nears the target position.

(4) Switch to the zero velocity stabilization law to stop the hovercraft.

(5) Turn in place until the desired heading is attained.

(6) If error tolerances are not met, go back to step (1). Otherwise, switch to the zero velocity stabilization law.

In theory, this control scheme should stabilize the origin of the full dynamics. The algorithm is inefficient, however, in that it does not take advantage of the absence of nonholonomic constraints that one encounters in wheeled robots. For example, it is not necessary to point the hovercraft before moving toward the target. A clever time-varying control law would use the kinematic freedoms to position the hovercraft in minimal time and with minimal energy.

Unfortunately, we have not yet identified a time-varying control law to stabilize the origin. In general, stabilizing the full-dynamics of underactuated vehicles requires geometric control concepts such as controllability, accessibility, and Lie Bracketing. In the next section, we address the nontrivial problem of pointing the hovercraft at a particular heading. We will use a stabilizability result proved by Brockett to illustrate the difficulty involved.

2.3.2 Heading Stabilization Difficulties

On the surface, heading stabilization might not seem difficult to achieve. After all, we have already demonstrated the existence of a control law to stabilize an arbitrary angular momentum (2.39). One might imagine that the addition of the kinematic state variable θ would not complicate matters too much. On the contrary, we will shortly argue why any attempt to find a smooth state feedback law for heading stabilization is futile.

Brockett's necessary condition for feedback stabilizability [8] provides a simple test to determine the existence of a continuously differentiable control law of the form $u = F(x)$ to stabilize a system equilibrium.

Theorem 2.3.3 (Brockett's Necessary Condition for Feedback Stabilizability) *Let* $\dot{x} = f(x, u)$ *be given with* $f(x_0, 0) = 0$ *and* $f(\cdot, \cdot)$ *continuously differentiable in a neighborhood of* $(x_0, 0)$. *A necessary condition for the existence of a continuously differentiable control law which makes* $(x_0, 0)$ *asymptotically stable is that:*

(i) the linearized system should have no uncontrollable modes associated with eigenvalues whose real part is positive.

(ii) there exists a neighborhood N *of* $(x_0, 0)$ *such that for each* $\xi \in N$ *there exists a control* $u_\xi(\cdot)$ *defined on* $[0, \infty)$ *such that this control steers the solution of* $\dot{x} = f(x, u_\xi)$ *from* $x = \xi$ *at* $t = 0$ *to* $x = x_0$ *at* $t = \infty$.

(iii) the mapping defined by

$$\gamma : (x, u) \longmapsto f(x, u) \tag{2.56}$$

should be onto an open set containing 0.

Note that while Theorem 2.3.3 provides a weaker result than the Theorem of Byrnes and Isidori, Brockett's Theorem does not require any additional hypotheses (other than the necessary conditions) to be satisfied. It is easy to show using a direct application of Brockett's Theorem that necessary condition (iii) is violated with the inclusion of θ in the state vector.

Proposition 2.3.4 *Given system dynamics* (2.10)*, together with* $\dot{\theta} = \frac{\Pi}{J}$*, there is no continuously differentiable state feedback law that stabilizes the equilibrium* $(P_X, P_Y, \Pi, \theta) = 0$*.*

Proof: We will consider (2.10) together with the equation $\dot{\theta} = \frac{\Pi}{J}$. These equations are independent of the remaining two equations in the full dynamics (2.11). Let η be the combined state and control vector given by $\eta = [P_X, P_Y, \Pi, \theta, F_X, F_Y]^{\mathrm{T}}$. We need to check whether the mapping, $\gamma : \eta \longmapsto f(\eta)$, is onto a neighborhood of the origin. Let $\boldsymbol{\alpha} \triangleq (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ be an arbitrary point in a neighborhood of $\gamma(\eta) = 0$. Now, using

$$\gamma(\eta) = \begin{bmatrix} P_Y \frac{\Pi}{J} + F_X \\ -P_X \frac{\Pi}{J} + F_Y \\ -d\, F_Y \\ \frac{\Pi}{J} \end{bmatrix} \tag{2.57}$$

and solving $\gamma(\eta) = \boldsymbol{\alpha}$, we immediately obtain:

$$\Pi = \alpha_4\, J \tag{2.58}$$

$$F_Y = -\frac{\alpha_3}{d} \tag{2.59}$$

$$-P_X\,(\alpha_4) - \frac{\alpha_3}{d} = \alpha_2 \tag{2.60}$$

$$P_Y\,(\alpha_4) + F_X = \alpha_1 \tag{2.61}$$

A problem occurs in equation (2.60) if $\alpha_4 = 0$. In this case, we obtain $\alpha_3 = -\alpha_2\, d$, which may be false depending on the arbitrarily chosen values α_2 and α_3. Thus, $\gamma(\eta) = \boldsymbol{\alpha}$ fails to be consistent for certain values of $\boldsymbol{\alpha}$, and we conclude that $\gamma(\eta)$

is not onto. The failure of condition (iii) in Theorem 2.3.3 implies that a $\mathcal{C}^1$ state feedback law stabilizing $(P_X, P_Y, \Pi, \theta) = 0$ does not exist. ■

Since Brockett's necessary condition is not satisfied, we should not waste our time searching for a continuously differentiable state feedback law to stabilize an arbitrary heading. Instead, we might consider time-varying feedback laws or a control algorithm that switches between two or more of the reduced dynamics laws discussed previously. It turns out that we can achieve fairly good performance by adopting a hybrid control approach and making use of control *primitives* to point the hovercraft at an arbitrary heading.

2.3.3 A Hybrid Approach to Heading Stabilization

At this point, two of the reduced dynamics control laws in particular should come to mind. Obviously, these are the zero velocity stabilization law (2.16) and the constant $\overline{\Pi}$ stabilization law (2.39). The most straightforward pointing algorithm consists of the following steps:

(1) Choose a pointing tolerance, $\delta > 0$, and a fixed angular momentum, $\overline{\Pi}$.

(2) Switch to the zero velocity stabilization law.

(3) Compute the error in alignment, $\hat{\theta} \triangleq \theta - \overline{\theta}$. If $|\hat{\theta}| \leq \delta$ go to (2).

(4) Wait until condition (2.48) is satisfied for the preselected $\pm\overline{\Pi}$.

(5) Switch to the $\overline{\Pi}$ stabilization law.

(6) Recompute $\hat{\theta}$. If $|\hat{\theta}| \leq \delta$, go to (2). Otherwise, wait for the pointing tolerance to be achieved.

Thus, the strategy is to switch between the two behaviors *stop* and *turn in place* as needed to maintain the desired heading $\overline{\theta}$. The two parameters to be chosen in this control scheme are δ and $\overline{\Pi}$. δ determines how precise we want the pointing to be, while $\overline{\Pi}$ controls the rate at which we get there. These parameters greatly affect the overall performance of the pointing algorithm. Choosing δ too small will cause the system to oscillate rapidly about the desired heading, wasting valuable energy. Picking $\overline{\Pi}$ too large will cause the hovercraft to overshoot the desired heading and also waste energy. Therefore, proper parameter values for a real system should be chosen using good design common sense or the aid of a simulator.

An extension to the heading stabilization algorithm proposed above is to vary the parameter value $\overline{\Pi}$ in response to the heading error. For example, define an error signal, $\hat{\theta}(t) \triangleq \theta(t) - \overline{\theta}$, representing the error in desired heading alignment. A simple proportional feedback control law is constructed by setting $\overline{\Pi}(t) = -k\,\hat{\theta}(t)$, where $k > 0$ is the gain. This control law captures the basic idea that for maximum performance, we should turn faster as we deviate farther from the desired heading. The angular velocity should continue to decrease as we approach the target, until a certain heading tolerance is achieved. At that point, the controller should switch to the zero velocity control law to stop the hovercraft and maintain the current heading.

The difficulty with this control scheme stems from the fact that the $\overline{\Pi}$ stabilization law can converge to two different limit points: $(0, 0, \overline{\Pi})$ and $(0, -md\frac{\overline{\Pi}}{J}, 0)$. Since initial conditions determine the equilibrium to which the system converges, care must be taken when varying $\overline{\Pi}$. Recall from section 2.2.5 the requirement to sufficiently damp all velocities before switching to the $\overline{\Pi}$ stabilization law. The condition states that smaller $\overline{\Pi}$ requires greater system "stillness" before switching control laws. Following the arguments presented in section 2.2.5, an equivalent restriction exists on the rate by which $\overline{\Pi}$ may be decreased.

We will now derive an inequality that relates the current $\overline{\Pi}$ and system velocities to a new *provisioned* $\overline{\Pi}^\star$. This newly computed $\overline{\Pi}^\star$ may then be substituted safely into the $\overline{\Pi}$ stabilization law. To fix ideas, let $W_v(\mathbf{x})$ refer to Lyapunov function (2.36), with $\hat{P}_X = P_X$, $\hat{P}_Y = P_Y$, and $\hat{\Pi} = \Pi - v$. Assume that we are currently running the $\overline{\Pi}$ stabilization law (2.39) and $W_{\overline{\Pi}}(\mathbf{x}) < \varepsilon$. First, note that

$$\begin{aligned} W_{\overline{\Pi}}(\mathbf{x}) < \varepsilon \quad &\Longrightarrow \quad \frac{\left(\Pi - \overline{\Pi}\right)^2}{2J} < \varepsilon \\ &\Longrightarrow \quad \left|\Pi - \overline{\Pi}\right| < \sqrt{2\varepsilon J} \end{aligned} \tag{2.62}$$

The local minimum of the Lyapunov function for a new $\overline{\Pi}^\star$ is obtained by evaluating $W_{\overline{\Pi}^\star}(\mathbf{x})$ at $(0, -md\frac{\overline{\Pi}^\star}{J}, 0)$. This yields:

$$W_{\overline{\Pi}^\star{}_{\text{bad}}} = \frac{\left(m\, d\, \overline{\Pi}^\star\right)^2}{2mJ^2} + \frac{\overline{\Pi}^{\star 2}}{2J} \tag{2.63}$$

We must now evaluate the Lyapunov function, $W_{\overline{\Pi}^\star}(\mathbf{x})$, at a general point $\mathbf{x}$. For notational simplicity, we replace the hat variables with their definitions.

$$\begin{aligned}
W_{\overline{\Pi}^\star}(\mathbf{x}) &= \frac{{P_X}^2 + {P_Y}^2}{2m} + \frac{\left(\Pi - \overline{\Pi}^\star\right)^2}{2J} \\
&= \frac{{P_X}^2 + {P_Y}^2}{2m} + \frac{\left(\Pi - \overline{\Pi} + \overline{\Pi} - \overline{\Pi}^\star\right)^2}{2J} \\
&= W_{\overline{\Pi}}(\mathbf{x}) + \frac{\left(\Pi - \overline{\Pi}\ \right)\left(\overline{\Pi} - \overline{\Pi}^\star\right)}{J} + \frac{\left(\overline{\Pi} - \overline{\Pi}^\star\right)^2}{2J} \\
&\leq W_{\overline{\Pi}}(\mathbf{x}) + \frac{\left|\Pi - \overline{\Pi}\ \right|\left|\overline{\Pi} - \overline{\Pi}^\star\right|}{J} + \frac{\left(\overline{\Pi} - \overline{\Pi}^\star\right)^2}{2J} \\
&< \varepsilon + \sqrt{2\varepsilon J}\,\frac{\left|\overline{\Pi} - \overline{\Pi}^\star\right|}{J} + \frac{\left(\overline{\Pi} - \overline{\Pi}^\star\right)^2}{2J}
\end{aligned} \tag{2.64}$$

where the last step follows from equation (2.62). We are now ready to derive the sufficient condition by setting $W_{\overline{\Pi}^\star}(\mathbf{x}) < W_{{\overline{\Pi}^\star}_{\text{bad}}}$ and using inequality (2.64):

$$W_{\overline{\Pi}^\star}(\mathbf{x}) \;<\; \varepsilon + \sqrt{2\varepsilon J}\,\frac{\left|\overline{\Pi} - \overline{\Pi}^\star\right|}{J} + \frac{\left(\overline{\Pi} - \overline{\Pi}^\star\right)^2}{2J} \;<\; \frac{\left(m\,d\,\overline{\Pi}^\star\right)^2}{2mJ^2} + \frac{{\overline{\Pi}^\star}^2}{2J} \tag{2.65}$$

We will assume first that $\overline{\Pi} - \overline{\Pi}^\star \geq 0 \implies 0 < \overline{\Pi}^\star \leq \overline{\Pi}$. Expanding and grouping terms in equation (2.65) yields the following inequality:

$$d^2 m\,{\overline{\Pi}^\star}^2 + 2J\left(\sqrt{2\varepsilon J} + \overline{\Pi}\right)\overline{\Pi}^\star - J\left(\sqrt{2\varepsilon J} + \overline{\Pi}\right)^2 > 0 \tag{2.66}$$

Immediately, we observe that there is a hope of satisfying inequality (2.66) since the l.h.s. is a convex function of $\overline{\Pi}^\star$. If we make (2.66) an equality, replace $\overline{\Pi}^\star$ with a dummy variable v, and determine that the equation admits real roots, then we need

only to choose a value for $\overline{\Pi}^{\star}$ larger than the max root. Solving for the max root, $v_{\max}$, yields:

$$\begin{aligned} v_{\max} &= \frac{-2J\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)+\sqrt{4J^2\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)^2+4d^2mJ\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)^2}}{2d^2m} \\ &= \frac{-2J\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)+\sqrt{4J^2\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)^2\left(1+\frac{d^2m}{J}\right)}}{2d^2m} \\ &= \frac{-J\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)+J\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)\sqrt{1+\frac{d^2m}{J}}}{d^2m} \\ &= \frac{J\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)\left(\sqrt{1+\frac{d^2m}{J}}-1\right)}{d^2m} \\ &= \frac{\sqrt{J}\left(\sqrt{2\varepsilon J}+\overline{\Pi}\right)\left(\sqrt{J+d^2m}-\sqrt{J}\right)}{d^2m} \end{aligned} \tag{2.67}$$

By choosing $0 < v_{\max} < \overline{\Pi}^{\star} \leq \overline{\Pi}$ we assure asymptotic convergence to the new angular velocity, $\overline{\Pi}^{\star}$. Conversely, if the assumption is made in (2.65) that $\overline{\Pi} - \overline{\Pi}^{\star} \leq 0$ $\implies \overline{\Pi} \leq \overline{\Pi}^{\star} < 0$, then the sufficient condition amounts to $\overline{\Pi} \leq \overline{\Pi}^{\star} < v_{\min} < 0$ where $v_{\min}$ is the min root given by:

$$v_{\min} = \frac{\sqrt{J}\left(\overline{\Pi}-\sqrt{2\varepsilon J}\right)\left(\sqrt{J+d^2m}-\sqrt{J}\right)}{d^2m} \tag{2.68}$$

Equations (2.67) and (2.68) may be combined into

$$|v| = \frac{\sqrt{J}\left(\sqrt{2\varepsilon J}+\left|\overline{\Pi}\right|\right)\left(\sqrt{J+d^2m}-\sqrt{J}\right)}{d^2m} \tag{2.69}$$

and the sufficient condition stated succinctly as:

$$0 < |v| < |\overline{\Pi}^{\star}| \leq |\overline{\Pi}|. \tag{2.70}$$

Using the results of the above analysis, the heading stabilization algorithm now takes the following form:

(1) Choose a pointing tolerance, $\delta > 0$, and proportional feedback gain, $k > 0$.

(2) Switch to the zero velocity stabilization law.

(3) Compute the error in alignment, $\hat{\theta} = \theta - \overline{\theta}$. If $|\hat{\theta}| \leq \delta$ go to (2).

(4) Compute the desired angular momentum, $\overline{\Pi}$, using the feedback control law, $\overline{\Pi} = -k\,\hat{\theta}$.

(5) Wait until condition (2.48) is satisfied for the selected $\overline{\Pi}$.

(6) Switch to the $\overline{\Pi}$ stabilization law.

(7) Recompute $\hat{\theta}$. If $|\hat{\theta}| \leq \delta$ go to (2).

(8) Evaluate equation (2.69) for $|v|$. Set $\overline{\Pi}^{\star} = \pm \max\left(|v|, k|\hat{\theta}|\right)$.

(9) Set $\overline{\Pi} = \overline{\Pi}^{\star}$ and go to (6).

This algorithm adds additional complexity to the bang-bang type control scheme originally proposed. Only simulation will allow us to conclude which hybrid controller performs best in terms of transient response, settling time, and steady-state error.

Chapter 3

Inertial Navigation Systems

3.1 Introduction

Consider a point mass moving along a one-dimensional line. Given the time history of acceleration, the velocity may be computed by integrating the acceleration with respect to time. Furthermore, the position of the point mass at any instant of time is given by an additional integration. Mathematically, the kinematic equations for the point mass are:

$$\begin{aligned} v(t) &= v(t_0) + \int_{t_0}^{t} a(\tau) d\tau \\ x(t) &= x(t_0) + \int_{t_0}^{t} v(\tau) d\tau \end{aligned} \tag{3.1}$$

where a is the acceleration, v is the velocity, and x is the position. The point mass system described above is an example of a one-dimensional *inertial navigation system* (INS) and requires measurements from only one accelerometer to determine velocity and position along a line.

Let us now extend this example and allow the point mass to move freely in two or three dimensions. The acceleration experienced by the particle has both magnitude and direction and must be treated as a vector. As before, perfect knowledge

of the acceleration vector time history allows us to compute the particle's velocity and position using the vector equations:

$$\begin{aligned} \mathbf{v}(t) &= \mathbf{v}(t_0) + \int_{t_0}^{t} \mathbf{a}(\tau) d\tau \\ \mathbf{x}(t) &= \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{v}(\tau) d\tau \end{aligned} \tag{3.2}$$

Practical inertial navigation systems are comprised of two main parts: an inertial measurement unit (IMU) and a processor to interface with the IMU and solve the navigation equations. As its name implies, an IMU is a device consisting of one or more inertial sensors (e.g. accelerometers, inclinometers, and gyroscopes) that provides information about the motion of a physical body with respect to an *inertial frame*. An inertial frame is a reference frame that is not rotating or experiencing any linear acceleration. It is also a reference frame in which Newton's Laws may be applied without any corrective terms, such as Coriolis force.[1]

Accelerometers are devices that measure the total acceleration (also called specific force) experienced by a body with respect to an inertial frame. The measured acceleration consists of two components – pure inertial acceleration and Earth's gravity. An accelerometer can not differentiate between these two acceleration components.

Gyroscopes measure angular velocity about a sensitive axis with respect to an inertial frame. Given perfect measurements, these two types of sensors provide all the information needed to determine the velocity, position, and orientation of

[1]Coriolis force is an example of a *fictitious force* that appears in dynamical equations written in a rotating reference frame.

a body in three dimensions at each point in time, assuming initial conditions are known. To see how these sensors are used in an INS, it is essential to understand the concept of reference frames.

3.1.1 Reference Frames

The position, velocity, and orientation of a body is meaningful only when specified with respect to a reference frame. Generally, a set of coordinate axes called the *body frame* is attached to the center of mass of a rigid body. The exact point of attachment is not essential, although selecting the center of mass greatly simplifies the kinematic equations. Several conventions exist for aligning the coordinate frame with the body. We will use the convention for aircraft, in which the x axis points in the direction of the nose, the y axis points in the direction of the right wing, and the z axis points toward the bottom of the aircraft.

There are also several different *navigation reference frames*. The choice of frame strongly depends on the specific application requirements. Some reference frames are static while others move along the surface of the Earth. The most common navigation reference frames include the Earth-centered Earth-fixed (ECEF) frame, the Earth geographic frame, the Earth geocentric frame, and the local tangent plane. The ECEF frame is unique among the reference frames in that it is an inertial frame.

3.1.2 Reference Frame Rotations

A theorem due to Euler states that for two coordinate systems in $\mathbb{R}^3$ sharing a common origin, there exists a unique vector in $\mathbb{R}^4$ that may be used to bring the first coordinate system into coincidence with the second. The vector, called a *quaternion*, prescribes the unique axis of rotation in $\mathbb{R}^3$ and the amount of rotation needed. When quaternions are constrained to have unit norm, they are elements of the generalized unit sphere in $\mathbb{R}^4$. Although quaternions have one more parameter than necessary to represent orientation in three dimensions, they are completely free from the singularities that plague other representations of attitude.

The information contained in a quaternion may also be packaged in matrix form. For example, a *Direction Cosine Matrix* (DCM) is an orthonormal matrix that belongs to SO(3), the Lie group that describes rotations in three-dimensional space. Given two coordinate systems related by a rotation, the columns of the DCM are the basis vectors of the first coordinate system resolved in the second coordinate system. Thus, pre-multiplying a vector in the first coordinate system by the DCM transforms the vector to the second coordinate system. Note that vector transformations between reference frames always preserve the norm of the transformed vector.

A third way to specify rotations is with three *Euler angles*. These angles have special names – yaw is rotation about the z axis, pitch is rotation about the y axis, and roll is rotation about the x axis. Though physically intuitive, Euler angles suffer from two major problems. First, the orientation resulting from three Euler rotations

is dependent on the order in which the individual rotations are executed. Although certain conventions exist (such as yaw-pitch-roll), there is always the potential for ambiguity. More importantly, Euler angles exhibit singularities at pitch angles of $\pm 90°$. At these singularities, the roll and yaw angles change instantaneously by $180°$.

We can now appreciate the importance of reference frames to inertial navigation systems. Measurements taken by the IMU in the body frame must be resolved in the navigation frame prior to updating the kinematic equations (3.2). Thus, accurate determination of the time-varying rotation matrix is crucial to the overall accuracy of the complete navigation solution.

3.1.3 Rotation Matrix Dynamics

For an inertial navigation frame, the evolution of the direction cosine matrix, B, satisfies the differential equation:

$$\dot{B}(t) = B(t)\,\hat{\Omega} \tag{3.3}$$

where $\hat{\Omega}$ is the skew symmetric angular velocity matrix given by $[\boldsymbol{\omega}\times]$. The vector, $\boldsymbol{\omega}$, denotes the angular velocity of the body frame with respect to the navigation frame, expressed in body frame coordinates.

A closed form solution [11] to equation (3.3) at time t_k is given by:

$$B(t_k) = \left[\boldsymbol{I} + \frac{\sin\left(\|\boldsymbol{v}\|\right)}{\|\boldsymbol{v}\|^2}\,\Gamma + \frac{1-\cos\left(\|\boldsymbol{v}\|\right)}{\|\boldsymbol{v}\|^2}\,\Gamma^2\right] B(t_{k-1}) \tag{3.4}$$

where $\Gamma(t_k) = [\boldsymbol{v}(t_k)\times]$, $v_i(t_k) = -\int_{t_{k-1}}^{t_k} \overline{\omega}_i(\tau)d\tau$ for piecewise constant $\overline{\omega}_i(t), t \in [t_{k-1}, t_k]$, and $i = 1, 2, 3$. Typically, a triad of gyroscopes is used to measure the

angular velocities about the three body axes. Equation (3.4) is well defined as $\|\boldsymbol{v}\| \to 0$, but care must be taken to ensure that the rotation matrix remains orthonormal after each update.

The rotation matrix dynamics for a two-dimensional INS simplify greatly. For a right-handed body frame, $\hat{\Omega}$ is the skew symmetric angular velocity matrix given by:

$$\hat{\Omega} = \begin{bmatrix} 0 & -\Omega \\ \Omega & 0 \end{bmatrix} \tag{3.5}$$

where Ω is the angular velocity about the body Z axis. Consequently, equation (3.3) reduces to the dynamics of a harmonic oscillator with time-dependent natural frequency, Ω, and solution at time t_k given by:

$$B(t_k) = \begin{bmatrix} \cos(\theta(t_k)) & -\sin(\theta(t_k)) \\ \sin(\theta(t_k)) & \cos(\theta(t_k)) \end{bmatrix} \tag{3.6}$$

where $\theta(t_k) = \theta(t_{k-1}) + \int_{t_{k-1}}^{t_k} \omega_Z(\tau) d\tau$ is the computed yaw angle, obtained by integrating the Z-axis angular velocity over the interval $[t_{k-1}, t_k]$. Thus, for a planar INS with the body XY plane parallel to the inertial xy plane, only a single gyroscope is needed to compute the orientation matrix.

3.1.4 Sensors and Technology

3.1.4.1 Accelerometers

Accelerometers are devices that measure the specific force vector. In principle, an accelerometer consists of a small proof mass attached to a damped spring system.

When the accelerometer is subjected to linear acceleration (or Earth's gravity) along its sensitive axis, the mass is displaced according to Newton's Second Law of Motion. The position of the proof mass satisfies the following differential equation [11]:

$$\ddot{\mathbf{r}} = -\frac{k}{m}\mathbf{r} - \frac{d}{m}\dot{\mathbf{r}} + \mathbf{G}(\mathbf{r}) \tag{3.7}$$

where $\mathbf{r}$ is the position of the proof mass, $\mathbf{G}(\mathbf{r})$ is the position-dependent gravitational field vector, k is the spring constant, d is the damping coefficient, and m is the mass. If we assume that the output of the accelerometer is given by $\mathbf{f} = -\frac{k}{m}\mathbf{r}$, then we obtain the following equation:

$$\alpha\dot{\mathbf{f}} = -\mathbf{f} + \ddot{\mathbf{r}} - \mathbf{G}(\mathbf{r}) \tag{3.8}$$

where $\alpha = \frac{d}{k}$ is the accelerometer bandwidth. For frequencies well within the bandwidth, the accelerometer output is given by:

$$\mathbf{f} = \ddot{\mathbf{r}} - \mathbf{G}(\mathbf{r}) \tag{3.9}$$

We see that the specific force measurement is a combination of both the pure linear acceleration experienced by a body and the local gravity vector. The gravity vector must be subtracted from the specific force measurement before using the measurement in the navigation equations. Fortunately, accurate models of the Earth's gravitational field exist and may be used to estimate the gravity vector at any location near the surface of the Earth.

3.1.4.2 Gyroscopes

Gyroscopes are sensors that measure angular velocity about a sensitive axis. The original single degree of freedom (SDF) mechanical gyroscope was pioneered by C.S. Draper at the Instrumentation Laboratory at the Massachusetts Institute of Technology. One of the first applications of the gyroscope was to stabilize Navy antiaircraft gunsights.

The functionality of the SDF gyroscope is based on the principle of *gyroscopic precession*. Gyroscopic precession occurs when a spinning wheel is acted on by a torque orthogonal to the spin axis. This applied torque is transferred to an axis perpendicular to the plane formed by the spin and torque axes. If the spinning wheel is mounted to a gimbal (so that the precession axis is free to rotate), the wheel will precess about this axis in response to an applied torque. The SDF gyroscope uses an angular pickoff device to measure the amount of gyroscopic precession about the output axis.

Physics dictates that an angular velocity, ω_i, about the gyro input axis produces a torque along the output axis. This torque causes the output axis to rotate at the rate $\omega_o = \frac{H}{C}\omega_i$, where H is the gyro angular momentum and C is the gyro viscous damping coefficient [12]. Thus, the angular position of the output axis is directly proportional to the integral of the input angular velocity.

In effect, the angular pickoff sensor in the SDF gyroscope provides a measurement of the integrated input angular velocity. The orientation of the body about the gyroscope's sensitive input axis may then be computed by appropriately scaling

the measurement. For this reason, the SDF gyroscope is referred to as an *integrating gyroscope*. Alternatively, an electric torque generator may be connected to the gyro output axis to oppose the precession described above. Knowledge of the applied torque (proportional to the electric current) may then be used to compute the angular velocity, w_i.

Through the years, several new gyroscope technologies have emerged. Common gyroscope technologies include the electrostatic gyro, the ring laser gyro, and resonator gyros. Each of these technologies takes advantage of different physical phenomena to measure angular velocity about a sensitive axis.

3.1.5 Shortcomings of Inertial Navigation Systems

Given perfect measurements of the specific force and angular velocity vectors, an inertial navigation system provides exact instantaneous velocity, position, and orientation according to equations (3.2) and (3.4). Inertial sensors are physical devices, however, and can provide measurements of only limited accuracy. While each measuring device has its own particular sources of error, there are several commonalities, including [13]:

- A fixed bias which may be corrected (usually by the manufacturer)
- A temperature dependent time-varying bias which may be corrected through sensor calibration
- A random bias that is constant throughout a run, but varies from turn-on to turn-on

- A time-varying in-run bias

Depending on the particular sensor, there may be additional sources of error that can be corrected through calibration. Sources of error common to both accelerometers and gyroscopes include:

- Scale Factor Errors – Scale factor errors are expressed as a ratio that relates a change in the output signal to a change in the physical quantity being measured.

- Cross-Coupling Errors – These errors result from sensitivity to an axis that is normal to the sensor axis.

- Frequency Dependent Errors – Each sensor has a finite bandwidth. Excitation at frequencies near the frequency cutoff results in erroneous measurements suffering from scaling errors and delay.

Each of these errors degrades the accuracy of an inertial navigation system. In particular, an accelerometer with a static random bias produces errors that grow linearly in the computed velocity and quadratically in position. Gyroscopes also typically exhibit a small bias, resulting in orientation errors that grow linearly. Left unchecked, these errors will increase without limit. One way to improve the accuracy of inertial navigation systems is by *aiding* the INS with additional information.

3.1.6 Aided Inertial Navigation Systems

An aided inertial navigation system consists of an IMU and additional measurements from a system that may be onboard or external to the INS. For example, an INS may use an onboard Doppler radar system to estimate the instantaneous velocity. On the other hand, the Global Positioning System (GPS) is an example of an external aiding system.

For an aided INS to perform well, the aiding device must sufficiently augment the amount of information available. The most effective aiding sensors possess characteristics that are complementary in nature to inertial sensors. For example, inertial sensors provide high bandwidth measurements, but suffer from random biases. In contrast, effective aiding sensors typically provide long-term stability at the expense of a lower update rate.

A well designed INS should combine all of the available information to form the best estimate of the system's orientation, velocity, and position. A Kalman Filter is often used to combine the measurements and provide a good estimate of the navigation state.

3.2 Problem Statement

3.2.1 The Need for an INS

The need for an accurate navigation system is motivated by the desire to achieve high performance closed loop control of a small radio controlled hovercraft.

Accurate linear and angular velocity estimates are needed for feedback control of the reduced dynamics (see section 2.2), while accurate position estimates are needed for trajectory tracking.

Unlike wheeled robots which may use odometry to estimate velocity, heading, and position, a hovercraft lacks any built-in system for measuring its navigation state. Therefore, we must develop an accurate aided INS to make such control efforts possible. The theory of inertial navigation and aided navigation systems is well understood. We found [11–13] to be excellent resources on INS theory, technology, and implementation issues.

3.2.2 Design Assumptions

Prior to undertaking the development of an INS, it is imperative to establish a set of specific performance requirements as well as realistic design and operational assumptions. If large deviations from the nominal operating conditions occur, we should not expect the INS to provide an accurate navigation solution.

Planar Assumption

We will assume that the hovercraft body is confined to the plane and thus normal to the gravity vector. Since the hovercraft is a planar vehicle, we will devote our effort to the development of a two-dimensional aided INS. Our decision to limit the navigation solution to the plane is strengthened by several practical and theoretical considerations explained below.

First, updating the rotation matrix is much easier for the planar case. Instead of evaluating equation (3.4) at each time step, we need only to integrate the output from the Z gyroscope and substitute the result into equation (3.6). Observe that (3.6) *guarantees* an orthonormal matrix for all time t_k by construction.

Determining attitude is also more difficult in the full three-dimensional setting. Initial attitude determination is usually accomplished by using the accelerometers to measure pitch and roll and a magnetometer to measure yaw. The IMU must remain stationary during the alignment process, as any uncompensated linear acceleration will corrupt the pitch and roll measurements. Accurate attitude measurement under dynamic conditions is more complex. Traditional methods for attitude determination include deterministic constrained least-squares using multiple vector observations and Extended Kalman Filtering. Novel approaches use advanced techniques such as adaptive filtering combined with traditional Kalman Filtering, Dynamic Programming, and state-matrix Kalman Filtering [14].

A deterministic approach to attitude determination stems from the work of Wahba [15]. In 1965, Wahba addressed the following problem: given two sets of vectors, each containing at least two elements that are nonzero and non-collinear, find the unique rotation matrix that achieves the best least-squares alignment between the two vector sets. For example, suppose that the IMU features a 3-axis magnetometer to measure Earth's magnetic field resolved in the body frame. Additionally, assume that the IMU is stationary so that the accelerometers provide a measurement of the gravity vector in body coordinates. If the gravity and magnetic field vectors are computed in the local navigation frame (using mathematical models

of Earth's local magnetic field and gravity), then the unique direction cosine matrix may be determined through an iterated least-squares approach.

It can be particularly challenging to determine three-dimensional attitude while accelerating. As discussed in section 3.1.5, gyros are prone to random bias errors. In many INS systems, the gyros are bias compensated using magnetometer measurements and low-pass filtered accelerometer data. A constrained least-squares approach, such as the q-method [16], is used to compute an estimate of the attitude for aiding purposes. For these Attitude/Heading Reference Systems (AHRS), it is assumed that the average linear acceleration over long time intervals is zero. Under this assumption, low-pass filtering removes the high frequency acceleration transients and yields an estimate of the gravity vector.

The problem is that there may be significant positively (or negatively) biased linear acceleration during prolonged vehicle maneuvers. In these situations, the filtered measured acceleration will not approximate the gravity vector, and the INS may develop large attitude errors. For example, attitude errors would likely occur while executing a prolonged turn or accelerating to speed. Although the INS would eventually correct these errors once the pure acceleration component was removed, convergence to the correct solution might take some time.

In recent work, researchers at Stanford University [17] demonstrated the success of a gyro-less attitude determination system on an actual airplane using a 3-axis magnetometer, a 3-axis accelerometer, and GPS velocity estimates. The team used time-differenced GPS velocity measurements as an estimate of the vehicle acceleration. This estimate was then subtracted from the accelerometer measurements to

obtain an estimate of the gravity vector in the body frame. The fusion of these attitude estimates with angular velocity measurements from a triad of gyroscopes would yield even better performance.

Each of the above approaches to three-dimensional attitude determination involves considerable complexity. Determining attitude for a two-dimensional INS is simplified greatly, as there is only one degree of rotational freedom. Thus, we need only to integrate the output of a single gyro to estimate the heading angle.

The two-dimensional INS assumption also simplifies the measurement of the vehicle's linear acceleration. Assuming that the planar hypothesis is valid, the X and Y-axis accelerometers will remain closely normal to the gravity vector and report only pure linear acceleration. Small perturbations in pitch and roll angle will inevitably contribute a small time-varying bias to each sensor. With proper aiding, however, we hope to estimate and remove these small biases in real-time to achieve an accurate navigation solution.

Inertial Reference Frame Assumption

We will use the tangent plane (North, East, Down) as the local navigation frame. The tangent frame is obtained by attaching a plane to the surface of the Earth at a specific point. Furthermore, we will assume that Coriolis acceleration may be neglected. Coriolis acceleration occurs when a body has nonzero velocity with respect to a rotating reference frame (the Earth). It is given by $\boldsymbol{\omega_e} \times \boldsymbol{v}$, where $\boldsymbol{\omega_e}$ is the Earth's angular velocity vector and $\boldsymbol{v}$ is the body velocity with respect to

the Earth. Since the hovercraft is a low-velocity vehicle (as opposed to a tactical missile), the error that results from neglecting Coriolis acceleration will be small.

3.2.3 Performance Goals

We wish to develop an aided two-dimensional INS suitable for control of a small hovercraft. The INS will provide bias corrected estimates of the Z-axis angular velocity and the x and y-axis linear acceleration. Additionally, the INS will provide accurate estimates of heading, linear velocity, and position, resolved in an inertial frame.

Aiding will be accomplished through the use of two auxiliary sensors. An indoor positioning system called "Cricket" [18, 19], similar in functionality to GPS, will provide two-dimensional position estimates to the INS. Additionally, an onboard magnetometer will provide estimates of the heading angle. We assume that any local magnetic field disturbances will be minimal and not adversely affect the heading determination. (Later, in section 5.4.2, we explain why this assumption is actually false.)

A Kalman Filter will be used to combine the sensor information and produce an estimate of the navigation state. In the next section, we discuss the theory of Kalman Filtering and its application to aided inertial navigation systems. A mathematically rigorous treatment of Kalman Filtering may be found in [20].

3.3 Kalman Filtering

3.3.1 Kalman Filtering Theory

The Kalman Filter is a recursive algorithm that combines multiple noisy observations to produce a minimum variance state estimate. When the system dynamics are linear and the disturbance is white Gaussian noise, the Kalman Filter is an optimal state estimator of the conditional mean, median, and mode [20]. As an optimal estimator, the Kalman Filter minimizes the mean square error of the state estimate.

The Kalman Filter was developed in the 1960s by Rudolf Kalman. One of the first Kalman Filtering applications was trajectory estimation for the Apollo missions. Since then, Kalman Filtering has found extensive use in aided navigation systems, where the fusion of multiple sensor measurements can improve navigation accuracy substantially.

We will now provide a brief overview of Kalman Filtering theory. In the following presentation, a sans-serif font is used to denote random variables.

Suppose we have a vector of measurements, $\mathsf{Z}(t_i)$, such that

$$\mathsf{Z}(t_i) \triangleq \begin{bmatrix} \mathsf{z}(t_1) \\ \mathsf{z}(t_2) \\ \vdots \\ \mathsf{z}(t_i) \end{bmatrix} \tag{3.10}$$

is comprised of vector measurements $\mathbf{z}(t_j)$ for $j = 1, 2, \ldots, i$. Assume further that the initial state vector $\mathbf{x}(t_0)$ is a Gaussian random variable and the state dynamics are linear and of the form:

$$\mathbf{x}(t_i) = \Phi(t_i, t_{i-1})\mathbf{x}(t_{i-1}) + B(t_{i-1})\mathbf{u}(t_{i-1}) + G(t_{i-1})\mathbf{w}(t_{i-1}) \tag{3.11}$$

where $\mathbf{x}(t_i) \in \mathbb{R}^n$, $\mathbf{u}(t_i) \in \mathbb{R}^m$, and $\mathbf{w}(t_i) \in \mathbb{R}^q$ is a zero-mean discrete white noise process with covariance given by:

$$E\left\{\mathbf{w}(t_i)\mathbf{w}(t_j)\right\} = \begin{cases} Q(t_i) & t_i = t_j \\ 0 & t_i \neq t_j \end{cases}. \tag{3.12}$$

Finally, assume that the measurement equation is of the form:

$$\mathbf{z}(t_i) = H(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{3.13}$$

where the measurement noise, $\mathbf{v}(t_i) \in \mathbb{R}^r$ is white and Gaussian with covariance:

$$E\left\{\mathbf{v}(t_i)\mathbf{v}(t_j)\right\} = \begin{cases} R(t_i) & t_i = t_j \\ 0 & t_i \neq t_j \end{cases}. \tag{3.14}$$

The Kalman Filter algorithm computes $\hat{\mathbf{x}}(t_i) \triangleq \mathrm{E}\left\{\mathbf{x}(t_i)|\mathbf{Z}(t_i)\right\}$ at each time step without actually storing the entire measurement history vector, $\mathbf{Z}(t_i)$. Since it is a recursive algorithm, the Kalman Filter greatly reduces memory and processing requirements and makes real-time computation of the state conditional mean possible. The filtering algorithm is comprised of two main steps: state update and state propagation. We will trace the algorithm from time t_i^- to time t_{i+1}^+, where the minus and plus superscripts denote time instants immediately before and after measurement incorporation respectively.

At time $t_i{}^-$, a new measurement, $\mathbf{z}(t_i)$ is available. The filter incorporates the measurement and updates the state estimate according to:

$$K(t_i) = P(t_i^-)H^{\mathrm{T}}(t_i)\left[H(t_i)P(t_i^-)H^{\mathrm{T}}(t_i) + R(t_i)\right]^{-1} \tag{3.15}$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + K(t_i)\left[\mathbf{z}(t_i) - H(t_i)\hat{\mathbf{x}}(t_i^-)\right] \tag{3.16}$$

$$P(t_i^+) = P(t_i^-) - K(t_i)H(t_i)P(t_i^-) \tag{3.17}$$

Equations (3.15) - (3.17) comprise the state update portion of the algorithm. $K(t_i)$ is the time-varying Kalman gain matrix and the term $\mathbf{z}(t_i) - H(t_i)\hat{\mathbf{x}}(t_i^-)$ is called the *innovation*. The gain matrix optimally weights the new information contained in the innovations to produce a new best estimate of the state. $P(t_i)$ is a positive definite matrix called the *error covariance matrix* and is defined by:

$$\begin{aligned} P(t_i) &= \mathrm{E}\left\{\mathbf{e}(t_i)\mathbf{e}(t_i)^{\mathrm{T}}\right\}, \\ \mathbf{e}(t_i) &\triangleq \mathbf{x}(t_i) - \hat{\mathbf{x}}(t_i) \end{aligned} \tag{3.18}$$

The term $\mathbf{e}(t_i)$ represents the amount of error in the state estimate at time t_i. Thus, $P(t_i)$ is an estimate of the amount of error present in the state estimate at each time step. Observe that the calculation of $K(t_i)$ and $P(t_i)$ does not depend on an actual realization of the measurement process, $\mathbf{z}(t_i)$. This property enables the gain and error covariance matrices to be precomputed offline and stored for later retrieval in a real-time application. Finally, the *measurement covariance matrix*, $R(t_i)$, is positive semidefinite and defined according to equation (3.14).

After the state update portion of the Kalman Filter algorithm has completed, the state $\mathbf{x}(t_i)$ and error covariance matrix $P(t_i)$ are propagated to time t_{i+1}^- according to the system dynamics model. The propagation equations are:

$$\begin{aligned} \hat{\mathbf{x}}(t_{i+1}^-) &= \Phi(t_{i+1}, t_i)\hat{\mathbf{x}}(t_i^+) + B(t_i)\mathbf{u}(t_i) \\ P(t_{i+1}^-) &= \Phi(t_{i+1}, t_i)P(t_i^+)\Phi^{\mathrm{T}}(t_{i+1}, t_i) + G(t_i)Q(t_i)G^{\mathrm{T}}(t_i) \end{aligned} \tag{3.19}$$

where $\hat{\mathbf{x}}(t_0)$ and $P(t_0)$ are the initial state and error variance matrix estimates respectively. Eventually, a new measurement becomes available, and the state estimate and error covariance matrix are updated to time t_{i+1}^+ using equations (3.15) - (3.17).

The state dynamics in (3.11) are linear in the state vector, which is assumed to be a Gaussian random variable. Furthermore, it is assumed that the input disturbance is white Gaussian noise. Since linear operations on Gaussian random variables preserve the Gaussian property, the state vector remains Gaussian throughout its evolution. In this special case, the Kalman Filter provides all the information needed to construct the conditional probability density function, $f_{\mathbf{x}(t_i)}\left(\mathbf{x}(t_i)|\mathbf{Z}(t_i)\right)$.

Additionally, when the dynamics are linear and the Gaussian noise assumption holds, the Kalman Filter produces a state estimate equivalent to $\mathrm{E}\left\{\mathbf{x}(t_i)|\mathbf{Z}(t_i)\right\}$. Since $\mathrm{E}\left\{\mathbf{x}(t_i)|\mathbf{Z}(t_i)\right\}$ is the optimal estimate of $\mathbf{x}(t_i)$ given the entire measurement history, $\mathbf{Z}(t_i)$, the Kalman Filter is the optimal state estimator in this case. If the Gaussian assumption does not apply, the Kalman Filter is still the best minimum error variance linear filter [20].

3.3.2 Kalman Filtering Applied to Inertial Navigation Systems

Aided inertial navigation systems combine measurements from multiple sensors to provide real-time estimates of velocity, orientation, and position. The challenge is determining how to combine these sensor measurements optimally to produce the best estimate of the navigation state.

We have seen that the Kalman Filter is the optimal state estimator when the dynamics are linear and the noise is white and Gaussian. The Kalman Filter is also an efficient recursive algorithm directly suited to real-time computation.

There are two main ways to incorporate Kalman Filtering into inertial navigation systems. In the *direct* implementation, the Kalman Filter is connected inline with the INS. For reasons that will soon be explained, the preferred alternative is the *indirect* implementation, in which the Kalman Filter forms a parallel branch with the INS.

3.3.2.1 Direct Implementation

The direct implementation is also called the *total error space* configuration. In this configuration, the Kalman Filter uses the actual navigation variables as states and implements a dynamical model of the INS. The inputs to the Kalman Filter are the raw sensor measurements from the IMU and the external aiding source(s). Figure 3.1 shows a diagram of the direct implementation. Perhaps not immediately obvious, there are several serious drawbacks to the direct implementation.

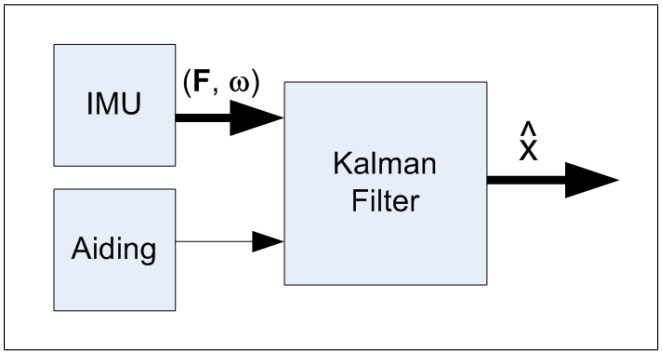


Figure 3.1: Direct Kalman Filter implementation. Thick arrows denote high data rate paths.

First, the INS dynamics model is nonlinear due to multiplication by the orientation matrix, B, in the navigation equations. In order to use Kalman Filtering, the dynamics must be linearized at each execution step. Second, the Kalman Filter must execute at the high data rate imposed by the inertial sensors. Propagating and updating the prediction equations at the high frequency IMU update rate requires large processing bandwidth and places undue burden on the CPU. The situation can be especially troublesome if the matrix to be inverted in the update step is large. Finally, there are practical issues regarding reliability in the direct implementation. If for some reason the Kalman Filter processor fails, the user will be unable to access any navigation estimates. It would be preferable to provide at least degraded navigation information while the Kalman Filter remained offline.

3.3.2.2 Indirect Implementation

A more robust and practical aided INS uses the indirect, or *error state space* implementation. The indirect implementation features the Kalman Filter block

in parallel with the INS. Rather than unnecessarily burdening the processor, the indirect implementation relies on the high bandwidth and short term stability of the INS to provide estimates of velocity, orientation, and position that are accurate over short intervals. The INS error drift rate is dependent on the quality of the inertial sensors.

Unlike the direct implementation which uses the navigation variables as states, the indirect Kalman Filter estimates the navigation *errors*. During the update step, the filter uses a measurement of the navigation error (obtained by subtracting the aiding measurement from the predicted value) to improve the estimate of the error states. Navigation error models are well understood, have low frequency dynamics, and are adequately modeled by linear systems [20]. The way in which the estimated error states are utilized depends on whether a feedback or feedforward configuration is used.

In the *feedforward* configuration (shown in Figure 3.2), there is no direct connection to the INS. The error states are subtracted from the time-integrated inertial measurements to form the corrected navigation outputs. The primary disadvantage of the feedforward implementation is the possibility for the error states to become large. Without any feedback, the INS integrators will continue to accumulate errors, and the errors states will grow without bound. Since the error dynamics usually are obtained through linearization of the system dynamics, the error states must be kept small in order for the linear model to remain valid.

Alternatively, the error states may be fed back to the INS and used to reset the integrators to corrected values. This *feedback* configuration (shown in Figure 3.3)

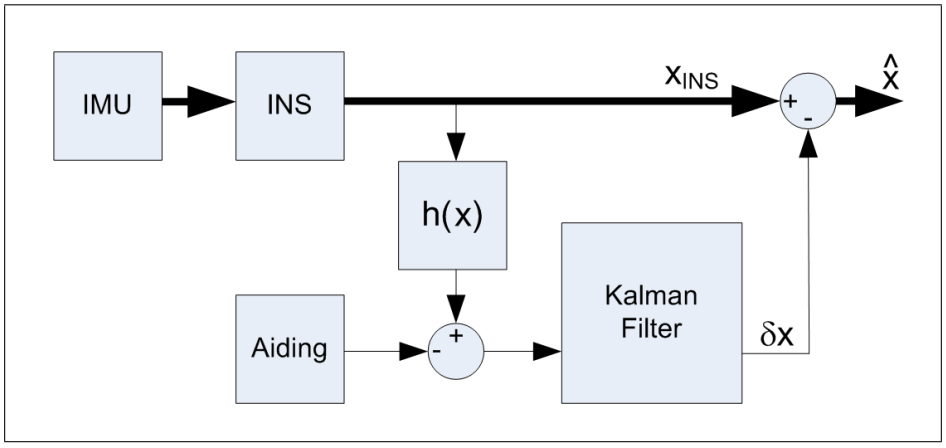


Figure 3.2: Indirect feedforward Kalman Filter implementation. Thick arrows denote high data rate paths.

prevents the navigation errors from growing too large, thus strengthening the validity of the linearized error model. In fact, the feedback configuration is essential for the *Extended Kalman Filter*, in which the dynamics are linearized at the navigation state estimate. For the Extended Kalman Filter, uncompensated errors can easily lead to filter divergence. An additional benefit provided by the feedback configuration is the trivial propagation of several of the error states between filter updates. At time t_i^+, the navigation errors have already been incorporated by the INS, resulting in a new state vector that has many, if not all, elements equal to zero.

3.3.3 INS Kalman Filter Design

In this section we describe the design of a planar aided INS using an indirect Kalman Filter implementation. First, we derive the linearized error dynamics model. Next, we describe a generic sensor error model that accounts for the primary sources of error. Finally, we present the complete error dynamics and measurement

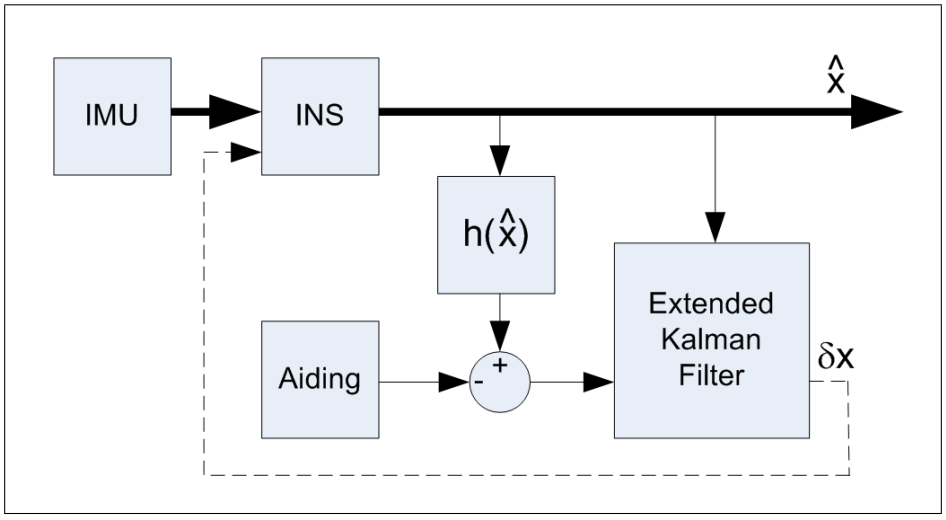


Figure 3.3: Indirect feedback Kalman Filter implementation. Thick arrows denote high data rate paths.

equations. The reader should refer to section 5.4.5 for information on the actual discrete error model implemented.

3.3.3.1 Error Dynamics Model

In this section, we derive the linearized error dynamics model for a two-dimensional INS, using an inertial navigation frame. The inertial frame is affixed to the surface of the Earth and is defined by the local North, East, and Down axes. We assume that the outputs provided by the navigation filter consist of the "true" quantity added together with an error term. We will use tilde variables to denote sensor measurements and hat variables to denote filter estimates.

Let us begin with a derivation of the heading error model. The heading error has the form:

$$\delta\theta(t) = \hat{\theta}(t) - \theta(t) \tag{3.20}$$

Differentiating both sides with respect to time yields:

$$\begin{aligned} \delta\dot{\theta}(t) &= \dot{\tilde{\theta}}(t) - \dot{\theta}(t) \\ &= \tilde{\omega}(t) - \omega(t) \\ &= \delta\omega(t) \end{aligned} \tag{3.21}$$

Thus, the heading error dynamics are linear. We now proceed to derive the velocity and position error dynamics. The velocity in inertial coordinates satisfies the following differential equation:

$$\dot{\mathbf{v}}(t) = B(\theta(t))\mathbf{F}(t) \tag{3.22}$$

where B is the rotation matrix from body to inertial coordinates and $\mathbf{F}(t)$ is the true specific force measurement. Observe that there is no need to account for gravitational acceleration since the gravity vector is normal to the navigation frame. The dynamics (3.22) are nonlinear for two reasons: (1) B is a nonlinear function of the navigation state θ, and (2) $B(\theta)$ multiplies the input, $\mathbf{F}(t)$. Following [21], we will linearize (3.22) in order to obtain the error dynamics model. Letting $g(\theta, \mathbf{F}) = B(\theta)\mathbf{F}$, we obtain:

$$\delta\dot{\mathbf{v}}(t) = \frac{\partial g}{\partial \theta}(\theta, \mathbf{F})\, \delta\theta(t) + \frac{\partial g}{\partial \mathbf{F}}(\theta, \mathbf{F})\, \delta\mathbf{F}(t) \tag{3.23}$$

Since we do not have knowledge of the true heading, θ, and specific force, $\mathbf{F}$, one alternative is to evaluate (3.23) at a precomputed nominal state trajectory. This is the basis for the *Linearized Kalman Filter*. The main problem with the Linearized Kalman Filter is that there is no guarantee that the nominal solution will match the

actual state trajectory. Often, better performance can be achieved by implementing an *Extended Kalman Filter*, in which the linearized dynamics are evaluated at the current state estimate and nominal input. Using the Extended Kalman Filter approach yields the following velocity error dynamics:

$$\delta\dot{\mathbf{v}}(t) = \begin{bmatrix} -\sin(\hat{\theta}) & -\cos(\hat{\theta}) \\ \cos(\hat{\theta}) & -\sin(\hat{\theta}) \end{bmatrix} \tilde{\mathbf{F}}(t)\,\delta\theta(t) + B(\hat{\theta})\,\delta\mathbf{F}(t) \tag{3.24}$$

or in component form:

$$\begin{aligned} \delta\dot{v}_x &= -\left(\tilde{F}_X \sin(\hat{\theta}) + \tilde{F}_Y \cos(\hat{\theta})\right)\delta\theta + \delta F_X \cos(\hat{\theta}) - \delta F_Y \sin(\hat{\theta}) \\ &= -\tilde{f}_y\,\delta\theta + \delta F_X \cos(\hat{\theta}) - \delta F_Y \sin(\hat{\theta}) \\ \delta\dot{v}_y &= \left(\tilde{F}_X \cos(\hat{\theta}) - \tilde{F}_Y \sin(\hat{\theta})\right)\delta\theta + \delta F_X \sin(\hat{\theta}) + \delta F_Y \cos(\hat{\theta}) \\ &= \tilde{f}_x\,\delta\theta + \delta F_X \sin(\hat{\theta}) + \delta F_Y \cos(\hat{\theta}) \end{aligned} \tag{3.25}$$

where the subscripts denote x and y vector components and the time argument has been suppressed for conciseness. Finally, the position error dynamics in inertial coordinates is given by:

$$\begin{aligned} \delta\dot{\mathbf{r}}(t) &= \dot{\hat{\mathbf{r}}}(t) - \dot{\mathbf{r}}(t) \\ &= \hat{\mathbf{v}}(t) - \mathbf{v}(t) \\ &= \delta\mathbf{v}(t) \end{aligned} \tag{3.26}$$

Combining equations (3.21), (3.25), and (3.26), we obtain the complete linear error dynamics model:

$$\left[\begin{array}{lcl} \delta\dot{\theta} & = & \delta\omega \\ \delta\dot{v}_x & = & -\tilde{f}_y\,\delta\theta + \delta F_X\cos(\hat{\theta}) - \delta F_Y\sin(\hat{\theta}) \\ \delta\dot{v}_y & = & \tilde{f}_x\,\delta\theta + \delta F_X\sin(\hat{\theta}) + \delta F_Y\cos(\hat{\theta}) \\ \delta\dot{r}_x & = & \delta v_x \\ \delta\dot{r}_y & = & \delta v_y \end{array}\right. \tag{3.27}$$

3.3.3.2 Sensor Error Model

We must provide an appropriate sensor error model for the error terms $\delta\omega$ and $\delta\mathbf{F}$ in equations (3.21) and (3.24). Rather than pursuing an error model that accounts for all known sources of error, including g-dependent biases, temperature biases, anisoelastic (g^2 dependent) biases, etc., we seek a simplified dynamics model that captures only the essential behavior.

For guidance in developing an appropriate error model, we refer to section 3.1.5, which outlines the main types of errors. A fixed or repeatable bias is easily modeled as a constant, and may even be compensated by the sensor manufacturer. Switch-on to switch-on variations are modeled as a random constant, while in-run variations may be modeled as Brownian motion driven by white noise of suitable strength [11].

If we lump the biases together, the following error model emerges for the Z-axis gyroscope:

$$\begin{aligned} \delta\omega(t) &= \mathsf{b}_\mathsf{g}(t) + \mathsf{w}_\mathsf{g}(t) \\ \mathsf{d}\mathsf{b}_\mathsf{g}(t) &= \mathsf{d}\beta(t) \end{aligned} \tag{3.28}$$

where $\mathsf{b}_\mathsf{g}(t)$ is the time-varying gyroscope bias modeled as a random walk, $\mathsf{d}\beta(t)$ is Brownian motion, and $\mathsf{w}_\mathsf{g}(t)$ is white Gaussian measurement noise. Similarly, the accelerometer error model is given by:

$$\begin{aligned} \delta\mathbf{F}(t) &= \mathbf{b}_\mathsf{a}(t) + \mathbf{w}_\mathsf{a}(t) \\ \mathsf{d}\mathbf{b}_\mathsf{a}(t) &= \mathsf{d}\boldsymbol{\beta}(t) \end{aligned} \tag{3.29}$$

where $\mathbf{b}_\mathsf{a}(t)$ is a vector of time-varying accelerometer biases modeled as a random walk and $\mathbf{w}_\mathsf{a}(t)$ is a vector of white Gaussian measurement noise. These models have proven sufficient for capturing the qualitative nature of the sensor errors.

3.3.3.3 Complete State Space Error Model

Observe that in (3.27), the heading error differential equation does not involve the velocity or position error states. Assuming the availability of heading aiding measurements to estimate the gyro bias, we may essentially treat heading determination and velocity/position estimation as separate problems. In this case, two separate Kalman filters may be implemented, with the output from the heading Kalman filter used as an input to the velocity/position filter.

Under the assumption that the heading estimate, $\hat{\theta}(t)$, provided by the heading filter closely tracks the true heading, we may approximate $\delta\theta(t)$ by zero in (3.24). The velocity error dynamics then simplify to:

$$\delta\dot{\mathbf{v}}(t) = B(\hat{\theta}(t))\,\delta\mathbf{F}(t) \tag{3.30}$$

where $\hat{\theta}(t)$ is provided by the heading Kalman Filter. For ease of implementation and additional system modularity, we will use separate Kalman Filters to implement the aided INS.

We now provide the complete error dynamics in matrix form. Combining equations (3.21) and (3.28), yields the following heading error model:

$$\begin{bmatrix} \delta\dot{\theta}(t) \\ \dot{\mathsf{b}}_{\mathsf{g}}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\theta(t) \\ \mathsf{b}_{\mathsf{g}}(t) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{w}_1(t) \tag{3.31}$$

where $\mathbf{w}_1(t)$ is a vector of white Gaussian noise. Combining equations (3.26), (3.29), and (3.30), we obtain the velocity and position error model:

$$\delta\dot{\mathbf{x}}_2 = A_2(t)\delta\mathbf{x}_2 + G_2(t)\mathbf{w}_2(t) \tag{3.32}$$

where

$$\delta\mathbf{x}_2(t) = [\delta \mathsf{v}_\mathsf{x} \;\; \delta \mathsf{v}_\mathsf{y} \;\; \delta \mathsf{r}_\mathsf{x} \;\; \delta \mathsf{r}_\mathsf{y} \;\; \mathsf{b}_{\mathsf{a}_\mathsf{x}} \;\; \mathsf{b}_{\mathsf{a}_\mathsf{y}}]^{\mathrm{T}}$$

$$A_2(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & \cos(\hat{\theta}(t)) & -\sin(\hat{\theta}(t)) \\ 0 & 0 & 0 & 0 & \sin(\hat{\theta}(t)) & \cos(\hat{\theta}(t)) \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$G_2(t) = \begin{bmatrix} \cos(\hat{\theta}(t)) & -\sin(\hat{\theta}(t)) & 0 & 0 \\ \sin(\hat{\theta}(t)) & \cos(\hat{\theta}(t)) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and $\mathbf{w}_2(t)$ is a vector of white Gaussian noise.

Measurement Equations

It is assumed that the external aiding sources provide a measurement, $\mathsf{z}(t)$, which consists of the true quantity corrupted by white Gaussian noise. As an example, consider a heading measurement obtained from an onboard magnetometer. The measurement error model is assumed to be:

$$\mathsf{z}_\theta(t) = \theta(t) - \mathsf{v}_1(t) \tag{3.33}$$

where $\mathsf{v}_1(t)$ is white Gaussian noise with appropriate power. If we now subtract this heading measurement from the INS measurement, obtained by integrating the Z-gyro, we obtain:

$$\begin{aligned} \delta \mathsf{z}_\theta(t) &\triangleq \hat{\theta}(t) - \mathsf{z}_\theta(t) \\ &= \delta\theta(t) + \theta(t) - (\theta(t) - \mathsf{v}_1(t)) \\ &= \delta\theta(t) + \mathsf{v}_1(t) \end{aligned} \tag{3.34}$$

The heading measurement equation takes the following form:

$$\delta \mathsf{z}_\theta(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \delta\theta(t) \\ \mathsf{b}_g(t) \end{bmatrix} + \mathsf{v}_1(t) \tag{3.35}$$

Similarly, a position measurement obtained using GPS or an alternative positioning system is assumed to have the following error model:

$$\mathbf{z}_r(t) = \mathbf{r}(t) - \mathbf{v}_2(t) \tag{3.36}$$

where $\mathbf{v}_2(t)$ is a two-dimensional vector of white Gaussian noise. Subtracting this position measurement from the INS position estimate, obtained by twice-integrating the accelerometer measurements yields:

$$\begin{aligned} \delta \mathbf{z}_r(t) &\triangleq \hat{\mathbf{r}}(t) - \mathbf{z}_r(t) \\ &= \delta\mathbf{r}(t) + \mathbf{r}(t) - (\mathbf{r}(t) - \mathbf{v}_2(t)) \\ &= \delta\mathbf{r}(t) + \mathbf{v}_2(t) \end{aligned} \tag{3.37}$$

The measurement equation in terms of the velocity and position error state variables is:

$$\delta \mathbf{z}_{\mathsf{r}}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta \mathsf{v}_{\mathsf{x}} \\ \delta \mathsf{v}_{\mathsf{y}} \\ \delta \mathsf{r}_{\mathsf{x}} \\ \delta \mathsf{r}_{\mathsf{y}} \\ \mathsf{b}_{\mathsf{a}_{\mathsf{x}}} \\ \mathsf{b}_{\mathsf{a}_{\mathsf{y}}} \end{bmatrix} + \mathbf{v}_2(t) \tag{3.38}$$

Thus, the complete heading error state space model is given by equations (3.31) and (3.35). Equations (3.32) and (3.38) give the complete velocity and position error state space model.

3.3.3.4 INS Equations

Theoretically, the INS updates its internal navigation variables according to the following set of equations:

$$\hat{\theta}(t_k) = \hat{\theta}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \tilde{\omega}(\tau) d\tau \tag{3.39}$$

$$\hat{\mathbf{v}}(t_k) = \hat{\mathbf{v}}(t_{k-1}) + \int_{t_{k-1}}^{t_k} B(\hat{\theta}(\tau)) \tilde{\mathbf{F}}(\tau) d\tau \tag{3.40}$$

$$\hat{\mathbf{r}}(t_k) = \hat{\mathbf{r}}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \hat{\mathbf{v}}(\tau) d\tau \tag{3.41}$$

We are limited, however, by the fact that the IMU provides discrete measurements at a sample rate of $\Delta t = t_k - t_{k-1}$. Thus, equations (3.39) – (3.41) must be replaced by

discrete integrators for actual implementation. For better accuracy, we interpolate the IMU measurements with line segments and compute the integrals directly:

$$\begin{aligned}
\hat{\theta}(t_k) &= \hat{\theta}(t_{k-1}) + \frac{\Delta t}{2}\left(\tilde{\omega}(t_k) + \tilde{\omega}(t_{k-1})\right) \\
\hat{\mathbf{v}}(t_k) &= \hat{\mathbf{v}}(t_{k-1}) + \frac{\Delta t}{2}\left(\tilde{\mathbf{f}}(t_k) + \tilde{\mathbf{f}}(t_{k-1})\right) \\
\hat{\mathbf{r}}(t_k) &= \hat{\mathbf{r}}(t_{k-1}) + \hat{\mathbf{v}}(t_{k-1})\Delta t + \frac{(\Delta t)^2}{6}\left(\tilde{\mathbf{f}}(t_k) + 2\tilde{\mathbf{f}}(t_{k-1})\right)
\end{aligned} \tag{3.42}$$

where $\tilde{\mathbf{f}}(t_k) = B(\hat{\theta}(t_k))\tilde{\mathbf{F}}(t_k)$.

3.3.3.5 Magnetometer Heading Determination

A magnetometer provides measurements of the Earth's magnetic field resolved in body coordinates. The Earth's magnetic field in the local tangent plane (North, East, Down) navigation frame is well known and can be readily obtained from the National Geophysical Data Center [22]. Knowledge of these two vectors enables heading to be determined in the navigation frame.

To compute heading, the measured and known magnetic field vectors, $\tilde{\mathbf{B}}$ and $\mathbf{b}$, are first projected onto the inertial frame. Unit vectors $\mathbf{u}_{\tilde{\mathbf{B}}_{proj}}$ and $\mathbf{u}_{\mathbf{b}_{proj}}$ are formed, and the heading angle magnitude is computed using:

$$|\mathsf{z}_\theta| = \left|\cos^{-1}\left(\left\langle \mathbf{u}_{\tilde{\mathbf{B}}_{proj}}, \mathbf{u}_{\mathbf{b}_{proj}} \right\rangle\right)\right| \tag{3.43}$$

with sign given by:

$$\operatorname{sgn}(\mathsf{z}_\theta) = \operatorname{sgn}\left\{\left(\mathbf{u}_{\tilde{\mathbf{B}}_{proj}} \times \mathbf{u}_{\mathbf{b}_{proj}}\right)_z\right\} \tag{3.44}$$

where, in equation (3.44), the sign is determined by the z component of the vector cross product.

3.3.3.6 Feedforward Kalman Filter Implementation

In section 3.3.3.3, we explained how the availability of heading aiding measurements allowed us to decouple the heading and velocity/position error equations in (3.27). The validity of this approach rested upon the assumption that $\hat{\theta}(t)$ was a good estimate of the true heading, $\theta(t)$, for all time t. Stated another way, we assumed a *certainty equivalence* between $\hat{\theta}(t)$ and $\theta(t)$.

Under this certainty equivalence hypothesis, the velocity and position error dynamics are truly linear (the heading error dynamics are also linear), and we may use a feedforward configuration for both Kalman Filters. If a heading aiding signal were not available, the coupled linearized error dynamics (3.27) would need to be implemented in a single Extended Kalman Filter using a feedback configuration.

The error state vectors, $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ at time t_0 are modeled as zero-mean Gaussian random variables:

$$\mathrm{E}\left\{\mathbf{x}_n(t_0)\right\} \equiv 0, \quad n = 1, 2 \tag{3.45}$$

and the initial error covariance matrices are given by:

$$P_1(t_0) = \begin{bmatrix} \sigma_{\delta\theta}(t_0) & 0 \\ 0 & \sigma_{\mathrm{b_g}}(t_0) \end{bmatrix} \tag{3.46}$$

for the heading error estimator and

$$P_2(t_0) = \begin{bmatrix} \sigma_{\delta \mathsf{v}_\mathsf{x}}(t_0) & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{\delta \mathsf{v}_\mathsf{y}}(t_0) & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{\delta \mathsf{r}_\mathsf{x}}(t_0) & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\delta \mathsf{r}_\mathsf{y}}(t_0) & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\mathsf{b}_{\mathsf{a}\mathsf{x}}}(t_0) & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\mathsf{b}_{\mathsf{a}\mathsf{y}}}(t_0) \end{bmatrix} \tag{3.47}$$

for the velocity and position error estimator. Theoretically, these values correspond to the error variances and sensor biases at time t_0. For our application, the actual values chosen are not critical for good filter performance, but should be nonzero.

The matrix Q in equation (3.12) captures the statistics of the noise processes driving the state evolution. For the heading error equations, $Q_1 \triangleq \mathrm{E}\left\{\mathbf{w_1}\mathbf{w}_1^{\mathrm{T}}\right\}$ is a 2×2 constant diagonal matrix. The noise covariance matrix for the velocity and position error dynamics is a 4×4 constant diagonal matrix given by $Q_2 \triangleq \mathrm{E}\left\{\mathbf{w_2}\mathbf{w}_2^{\mathrm{T}}\right\}$.

Examining the sensor error models for the gyroscope (3.28) and accelerometers (3.29), we see that each model has a white Gaussian measurement noise component. The noise powers for these processes occupy elements $(1,1)$ in Q_1 and $(1,1)$ and $(2,2)$ in Q_2. These values are most easily obtained by collecting sensor data over a long period of time with the IMU stationary. The sensor noise powers are then determined by computing the variance of the collected data.

Appropriate noise powers for the random walk processes are best obtained through simulation. Larger noise powers cause the filter to track variations in the sensor biases more quickly, but at the expense of increased steady-state error. Like-

wise, smaller noise powers sacrifice speed in bias tracking for smaller steady-state errors. In actual INS implementations, it is best to leave these values configurable so that the Kalman Filter can be tuned for optimal performance on-the-fly.

The measurement covariance matrices, R_1 and R_2, defined by equation (3.14), quantify the measurement noise power of the aiding sensors. Nominal values can be obtained by collecting stationary sensor data over a long period of time and computing the variance. For greater flexibility, these values may be left as parameters to be adjusted at run-time.

Part II

Physical Implementation

Chapter 4

The R/C Model Hovercraft

Our search for a suitable model hovercraft to test the control laws and INS led us to Germantown, Maryland, where Kevin Jackson, founder of www.hovercraftmodels.com, designs small radio-controlled hovercraft models. We purchased a fully-assembled HoverDemon Turbo hovercraft model for $200, which is a 15:1 ($3 \times 2$ feet) scale model of an actual passenger-carrying hovercraft that operates in St. Petersburg, Florida. In the following sections, we describe the mechanical design of the HoverDemon Turbo hovercraft and highlight the modifications we made for increased control authority and performance.

4.1 Design and Construction

The HoverDemon is made of a lightweight and durable corrugated plastic material. The embedded supports give the plastic material excellent rigidity and stiffness. A high density foam block forms the core of the vehicle and provides a solid foundation for the various mechanical and structural assemblies. So far, the hovercraft has survived several low-to-medium velocity collisions and has not shown any signs of deformation or other structural damage.

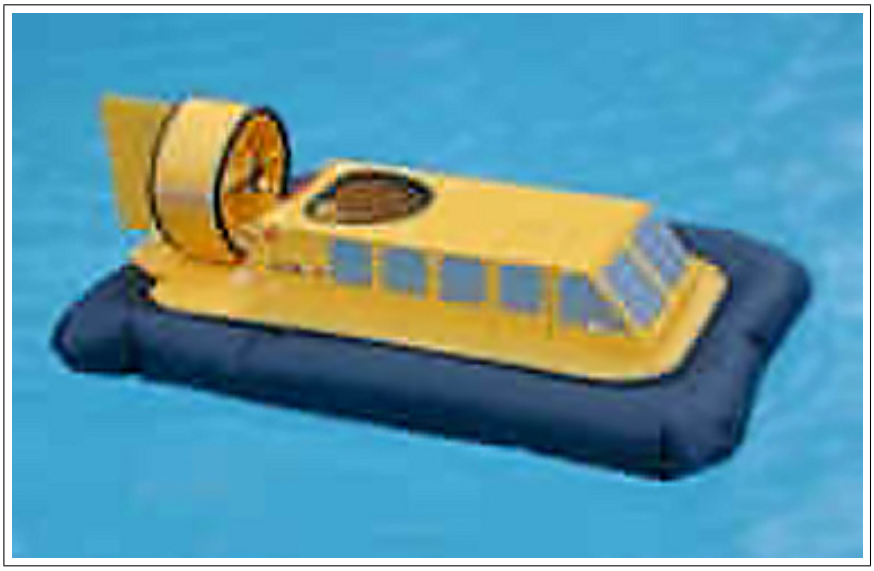
The skirt is quite durable and made from a lightweight vinyl material. It has endured many hours of operation and suffered only a few minor tears. We were able to mend the skirt in-place using a small amount of rubber cement.

The main structure of the hovercraft is a flat deck made from the corrugated plastic material. It is built on top of the foam core and supports the canopy and thrust duct assembly. The canopy is removable and encloses a small volume of space that is ideal for housing the batteries, radio receiver, and other supporting electronics. The thrust duct assembly is the heaviest structure on the hovercraft.

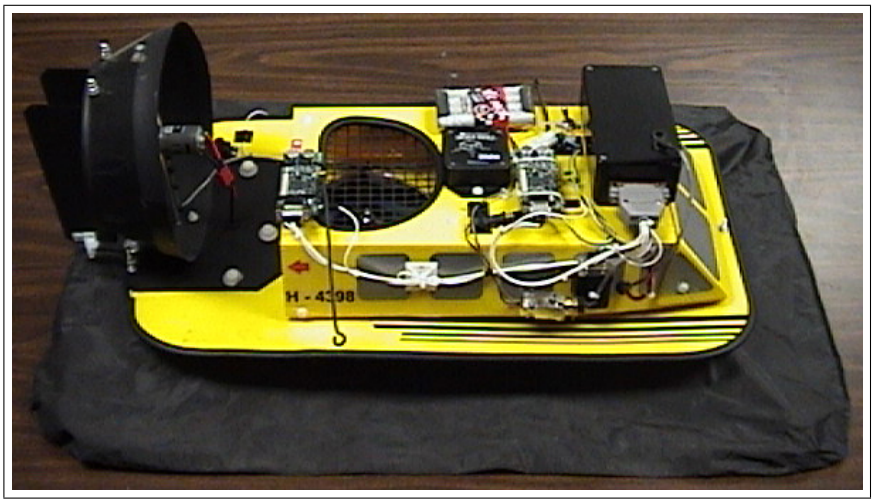
Figures 4.1a - 4.1c depict the HoverDemon in its stock form and after significant custom modification. Our modifications include:

- Replacement of the flimsy corrugated plastic duct with a larger and more rigid structure made from a plastic Tupperware container.
- Repositioning the thrust fan *inside* the thrust duct, with a tight clearance between the fan tips and the duct.
- Adding an additional rudder and designing a spring-loaded mechanical linkage to move the rudders in tandem.

An investigation of hovercraft design literature [23, 24] led to the modifications outlined above. Although each of these modifications required a great deal of time and effort, the end result was a substantial improvement in the vehicle control authority. Specifically, we achieved improved steerability and greater thrust production, especially in reverse. Since steering in reverse is always a challenge for vehicles with a rudder, the increase in performance was quite significant.

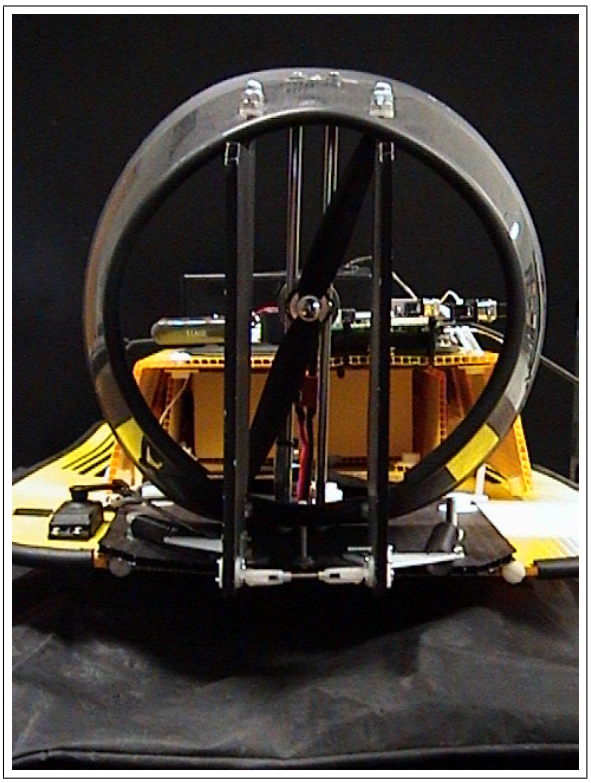


(a) Stock HoverDemon (Photo courtesy of www.hovercraftmodels.com)



(b) Modified HoverDemon

Figure 4.1: Comparison of the stock HoverDemon with the modified vehicle



(c) Modified HoverDemon (Thrust duct/rudder assembly)

Figure 4.1: Comparison of the stock HoverDemon with the modified vehicle

The increased thrust is extremely useful for braking the hovercraft quickly. In stock form, the feeble thruster was useless as a braking mechanism. In fact, full reverse power was only able to move the hovercraft backward at very low speed. The new design produces significant thrust at lower RPMs and is quite effective at propelling the hovercraft both forward and in reverse.

The HoverDemon is designed to carry 2-3 pounds of payload. The weight of the vehicle with all batteries and electronic subsystems in place is a staggering 5.7 pounds (2.6 kg)! We determined the moment of inertia[1] to be 2.3 lbs ft^2 (.096 kg m^2) and measured the distance from the hovercraft center of mass to the point of thrust production to be 12.6 inches (32 cm). The model hovercraft physical parameters are summarized below in table 4.1.

Mass (m)		**Lever Arm** (d)		**Moment of Inertia** (J)	
lbs	kg	inches	cm	lbs ft^2	kg m^2
5.7	1.6	12.6	32.0	2.3	.096

Table 4.1: R/C hovercraft physical parameters

4.2 Actuation

The two types of actuation systems present on the hovercraft are high-speed electric motors and a high-precision servo. The motors power the thrust and lift fans while the servo positions the rudder.

[1]See section 4.3 on system calibration.

4.2.1 Thruster and Lift Fan

The thrust and lift fans are driven directly by Speed 400 brushed electric motors. These motors are ubiquitous among R/C electric airplane hobbyists due to their small size, light weight and large power output. The motors we used are rated for 7.2 V and have a maximum stall torque of 12 oz-in. Nominally, they draw between 5 and 6 amps of current, but can peak as high as 8 amps. Their no-load speed is roughly 19,200 RPM, and we measured a max speed under load of approximately 12,000 RPM.

The thrust fan is a 2-bladed 3×7 plastic propeller designed for electric airplanes. This was a major upgrade from the 3.5 × 5 propeller supplied with the hovercraft. The smaller pitch (first number) and larger diameter (second number) allows the propeller to turn faster and produce a larger column of high velocity air. The net result is a substantial boost in maximum thrust. The lift fan is a three-bladed 6 inch plastic propeller. There was no need to modify the lift propeller.

An electronic speed controller (ESC) is needed to vary the speed of the electric motors. Without the ESC, the motor is either full-on or completely off. Most speed controllers achieve speed variation by adjusting the duty cycle of a high frequency (>1 kHz) square wave drive signal connected to the motor. Several of the high-end ESCs are reversible and feature an embedded microcontroller to configure the device and provide additional functionality. For example, on some ESCs, the maximum motor acceleration can be limited, and a programmable delay can be activated when

switching the motor from forward to reverse. The delay is useful for electric cars and trucks where a sudden transition to reverse could damage the transmission.

The hovercraft features two reversible ESCs. Out of the box, the HoverDemon includes a reversible ESC (DuraTrax 16T Mild-Modified) to control the thrust fan. The problem with this particular ESC is that the delay from forward to reverse can not be disabled. Since the propeller is coupled directly to the motor shaft, there is no need to delay when switching the motor direction. Additionally, since actuator delay decreases system stability, we desired to reduce the delay as much as possible.

The obvious solution was to replace the ESC with a different model. Much to our dismay, we discovered that all reversing speed controllers feature a small delay to prevent damage to connected mechanical components. We finally identified a speed controller (Tekin Rebel 2) which claimed the ability to completely disable the reverse delay. We purchased the unit, disabled the delay, and evaluated its performance. Unfortunately, a small delay of about 150 ms was still present. We determined that we could lessen the effect of the delay on the system performance by wiring the ESC in reverse. Since the hovercraft thruster is more effective producing forward rather than reverse thrust, switching the delay from reverse to forward helps to reduce the thruster bias. We also connected the original DuraTrax ESC to the lift fan in order to control the lift height.

We should note that neither of the ESCs described above uses any internal feedback loops to maintain speed under load. Since motor speed varies with applied voltage, the motor speed is a function of both the ESC output duty cycle and the battery voltage. Thus, for a constant ESC command, the motor speed will gradually

decrease as the battery voltage sags. We address this problem later in sections 4.3.2.1 and 5.2.

4.2.2 Rudder Servo

A Futaba hobby servo controls precise positioning of the rudder. The servo is controlled via pulse width modulation (PWM). A 50 Hz pulse with a variable pulse width between 1 and 2 ms is applied to the control input of the servo. The servo decodes the pulse width and moves the motor shaft to a corresponding location. (For example, 1 ms is full counter-clockwise and 2 ms is full clockwise). As long as the input does not change, the servo maintains the shaft in the same position via an internal feedback loop.

The servo is connected to the rudder via a thin metal rod. A control horn on the rudder converts the push/pull motion of the control rod into a torque. Another set of control horns is used to couple the two rudders so that their motion is synchronized. Finally, the rudders are spring loaded to prevent a singular mechanical configuration in which the rudders become locked. The springs also help remove mechanical hysteresis, leading to better repeatability in rudder positioning.

4.3 Calibration

A significant amount of time and effort was spent on an accurate vehicle calibration. Calibration is necessary to ensure that the forces requested by the con-

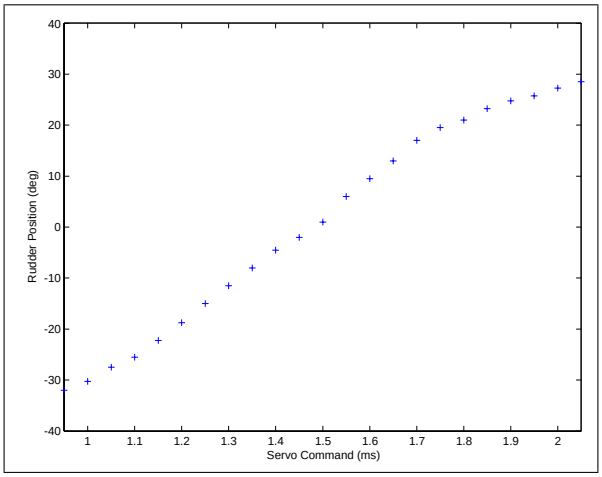


Figure 4.2: Rudder calibration data

troller are the same as the actual forces imparted by the actuators. The goal of the calibration was twofold:

(1) Derive a lookup table relating commanded rudder angle to servo position

(2) Quantify the relationship between propeller RPM and rudder angle to the net force produced

4.3.1 Rudder Calibration

Deriving the rudder lookup table was the easier of the two calibration tasks. Using a custom software GUI that we developed, we simply stepped the servo through its entire range (in steps of 50 μs) and recorded the rudder position for each command. A plot of rudder position vs servo command is shown in Figure 4.2. The relationship is nonlinear, and we use a simple lookup table with linear interpolation to model the dependence.

4.3.2 Thrust Calibration

Determining the relationship between propeller speed, rudder angle, and resultant force was not an easy task. Difficulties arose throughout the calibration, as we lacked the proper equipment to measure the body X and Y force components directly. Eventually, we devised a method to characterize the forward thrust production in terms of motor RPM and rudder angle.

4.3.2.1 Forward Thrust Calibration

Our approach was to tether the hovercraft with a force gauge at the point of force production. This configuration allowed the hovercraft to turn in place in response to a commanded rudder angle position and enabled us to record the resultant force directly. By measuring the resulting hovercraft angle, ψ, with respect to a fixed reference, the force components were given by:

$$F_X = \quad \|\mathbf{F}\| \cos(\psi) \tag{4.1}$$

$$F_Y = -\|\mathbf{F}\| \sin(\psi) \tag{4.2}$$

where $\|\mathbf{F}\|$ was the measured resultant force magnitude.

Obviously, test repeatability is of paramount importance since the controller must blindly trust the calibration during autonomous operation. Our first attempt at force calibration failed because we did not have a way to regulate the speed of the thrust fan during the calibration procedure. As the battery voltage decreased, so did the motor speed.

We realized that we needed a low-level controller onboard the hovercraft to regulate the thrust fan angular velocity. After several hours of searching, we identified a suitable miniature incremental encoder (US Digital, model E4, 100 CPR). Attaching the encoder disk and supporting electronics to the motor shaft proved to be quite difficult and required some special machining to achieve the required tolerances. We implemented a simple PI controller in firmware[2] and adjusted the gains through experimentation. Finally, we were ready to attempt calibration again.

To achieve accurate test results, we leveled a large table by placing metal shims under the legs. We used a spring-loaded force gauge with a maximum scale of 2 N and a resolution of 0.1 N. A straight line was marked on the table and used as a reference to keep the hovercraft tether cable aligned during calibration. The marked line also served as the angle reference for measuring the hovercraft's orientation, ψ. The test procedure consisted of the following steps:

(1) Command the rudder servo to the desired angle.

(2) Set the thrust fan to 5000 RPM (to keep the hovercraft from drifting off the table when the lift fan is turned on).

(3) Enable the lift fan. The lift fan speed should be set just slightly under the point where the skirt begins to flutter.

(4) Allow the hovercraft to seek its desired orientation while keeping the tether cable aligned with the marked line reference. Apply small disturbances to the

[2]See section 5.2 on the microcontroller system.

hovercraft so that the vehicle angle does not get stuck in a local minimum caused by friction at the tether point.

(5) Once the hovercraft has settled into its final orientation, add barriers on either side of the hovercraft as a pen.

(6) Decrease the thrust to 3000 RPM. This RPM corresponds to the minimum thrust that will register on the force gauge.

(7) Step up the thrust fan in increments of 1000 RPM. Record the resultant force for each setting.

(8) Once the thrust fan reaches its maximum RPM (dependent on battery voltage), cut the lift fan first, and then stop the thruster.

(9) Use chalk to mark the hovercraft body angle on the table.

(10) Measure the angle ψ formed by the chalk line and the marked line reference.

Once the data was collected, we created plots of resultant force vs thrust fan RPM for each rudder angle. As expected, the plots showed a quadratic relationship between force and RPM. We determined a best-fit quadratic equation for each data set and then used the models to create a large 2-D lookup table of motor RPMs. The 2-D lookup table was essentially a 14×81 matrix with the rows and columns representing equally-spaced rudder angles (5° increments) and force values (.025 N increments) respectively.

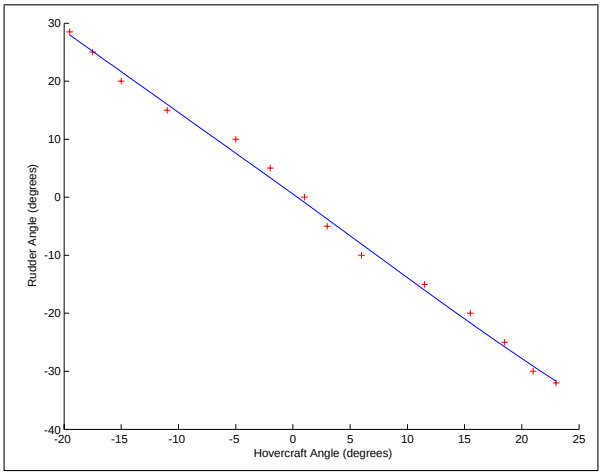


Figure 4.3: Rudder angle vs hovercraft (force vector) angle

We also plotted rudder angle vs hovercraft angle (shown below in Figure 4.3) and computed the following regression line:

$$\phi = -1.40\psi + .566^\circ \tag{4.3}$$

where ϕ is the rudder angle and ψ is the hovercraft angle in degrees. This relationship is needed by the controller to determine rudder angle given desired force components, F_X and F_Y. Observe that roughly 40% more rudder angle is needed for a desired hovercraft body angle due to aerodynamic losses in the rudder system.

4.3.2.2 Reverse Thrust Calibration

The calibration procedure described above is limited to measuring forward thrust only. We would need a compressive force gauge in order to calibrate reverse thrust using the same setup. Alternatively, attaching the tether cable to the nose of the hovercraft also would not solve the problem. For nonzero rudder angles, the torque produced would cause the hovercraft to continue turning.

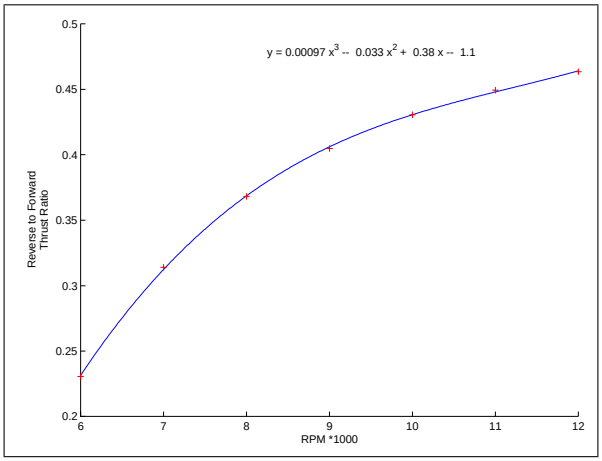


Figure 4.4: Reverse thrust calibration data

Our solution was to collect reverse thrust data with the rudder at the zero position only. We then modeled the relationship between the reverse-to-forward thrust ratio and motor RPM for the zero rudder case. We determined the best fit model to be a third-order polynomial shown in Figure 4.4 with equation:

$$y = 9.734\text{e-}4\, x^3 - 3.277\text{e-}2\, x^2 + 3.833\text{e-}1\, x - 1.099 \tag{4.4}$$

where x is the motor RPM (divided by 1000) and y is the reverse-to-forward thrust ratio. Motor RPMs for *arbitrary* reverse thrust values and nonzero rudder angles were computed using the polynomial model and the previously collected forward thrust data.

Using the derived force/RPM models, we generated the complete 14×161 2-D lookup table as well as a reverse force lookup table. The reverse lookup table provides resultant force as a function of motor RPM and rudder angle. It is depicted as a three-dimensional mesh below in Figure 4.5. Notice the quadratic dependence of resultant force on motor RPM.

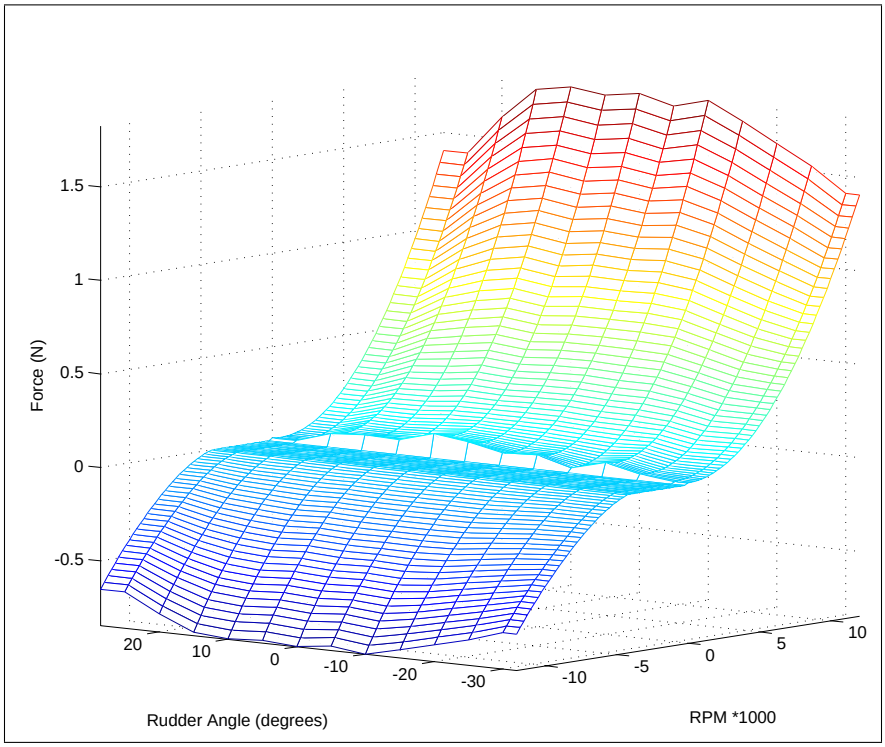


Figure 4.5: Reverse lookup table (RLUT) mesh plot. Force increases quadratically with motor RPM and decreases with rudder angle for a given RPM

4.3.3 Moment of Inertia Determination

We devised a simple method to determine the hovercraft moment of inertia using only the autopilot system hardware. Referring to the hovercraft dynamics (2.10), we see that the third equation relates the change in angular momentum to the Y-component of applied force. Since angular momentum Π is equal to the product of angular velocity Ω and the moment of inertia J, we can estimate the moment of inertia using the INS and the thrust calibration tables derived previously.

Data was collected to determine the moment of inertia using the following procedure:

(1) Place the hovercraft on a flat level surface.

(2) Set the rudder to an arbitrary nonzero angle using the rudder calibration table.

(3) Command a large force value (> 1 N) to overwhelm any small unmodeled friction.

(4) Start collecting data on the dSPACE system.

(5) Hold the hovercraft in place and activate the lift fan.

(6) Release the hovercraft.

(7) Cut the lift fan after about 5 seconds.

(8) Repeat for different rudder angles and thrust values.

For each trial, the INS angular velocity was plotted against time. Initially, each plot exhibited a linear relationship until frictional effects entered the dynamics. This

can be seen in Figure 4.6, which contains three trials for a rudder angle of -30°. We restricted the data set to the linear portion and computed the slope of the best-fit line. The moment of inertia for the ith trial was given by:

$$\hat{J}_i = -d\frac{F_Y}{\dot{\Omega}} \tag{4.5}$$

We averaged $\hat{J}_i$ obtained from several trials in order to form the best estimate of the moment of inertia, $\hat{J}$. We determined $\hat{J}$ to be .096 kg m^2. Evaluating the data collected, we observed that the best results were achieved using larger rudder angles and forces. Data for two of the trials appears below in Table 4.2.

	Rudder Angle (ϕ)	**Hovercraft Angle (ψ)**	$\|\mathbf{F}\|$	F_Y	$\dot{\Omega}$	$\hat{J}_i$
Trial 1	-20°	14.6°	1.1	-.28	.9412	.0952
Trial 2	25°	-17.4°	1.05	.31	-1.026	.0967

Table 4.2: Data collected to determine the moment of inertia

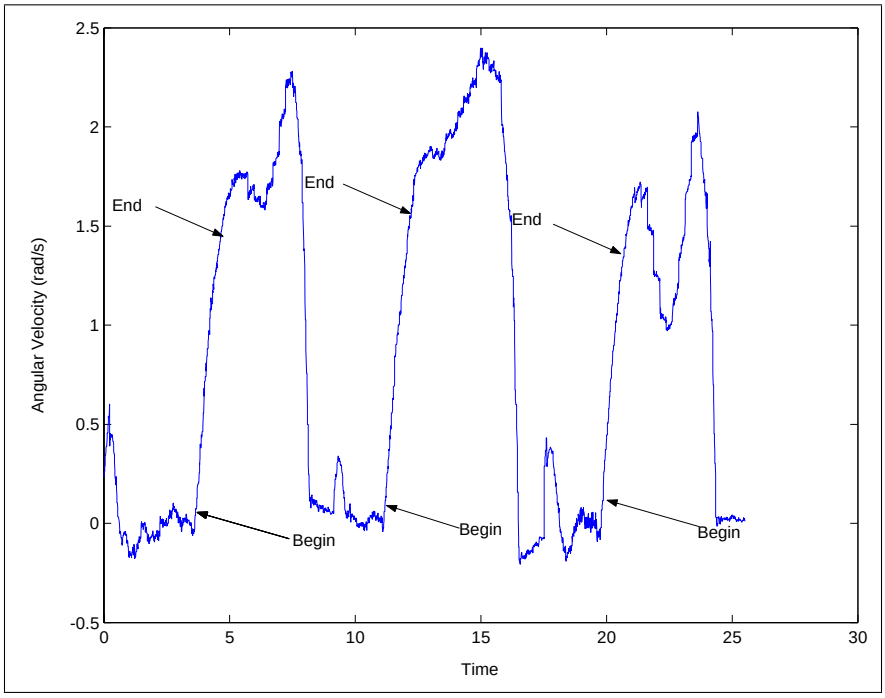


Figure 4.6: Experimental angular velocity data used to determine the moment of inertia. A force of 1.0 N was applied at a -30° rudder angle.

Chapter 5

The Autopilot

A block diagram of the autopilot system architecture appears in Figure 5.1. In the following sections, we will describe the functionality provided by each subsystem block.

5.1 dSPACE

The dSPACE system enables rapid prototyping of control systems. It consists of a dedicated processor running a proprietary real-time operating system (RTOS) and an interface board with extensive I/O capabilities for interacting with the physical world. Most importantly, dSPACE interfaces directly with Matlab and allows Simulink block diagrams to be compiled and run in real-time on the system. The main benefit to using dSPACE is that control systems may be simulated with Matlab, compiled to C code, and then evaluated in real-time on physical hardware without changing the underlying Simulink model.

dSPACE is the core processing system in the hovercraft autopilot. The system receives raw IMU data from the hovercraft over the Bluetooth link, executes the Kalman Filter and control law algorithms, and sends actuator commands back to the vehicle. As an RTOS, dSPACE provides the assurance that all computations finish within a specified amount of time and that discrete-time algorithms execute

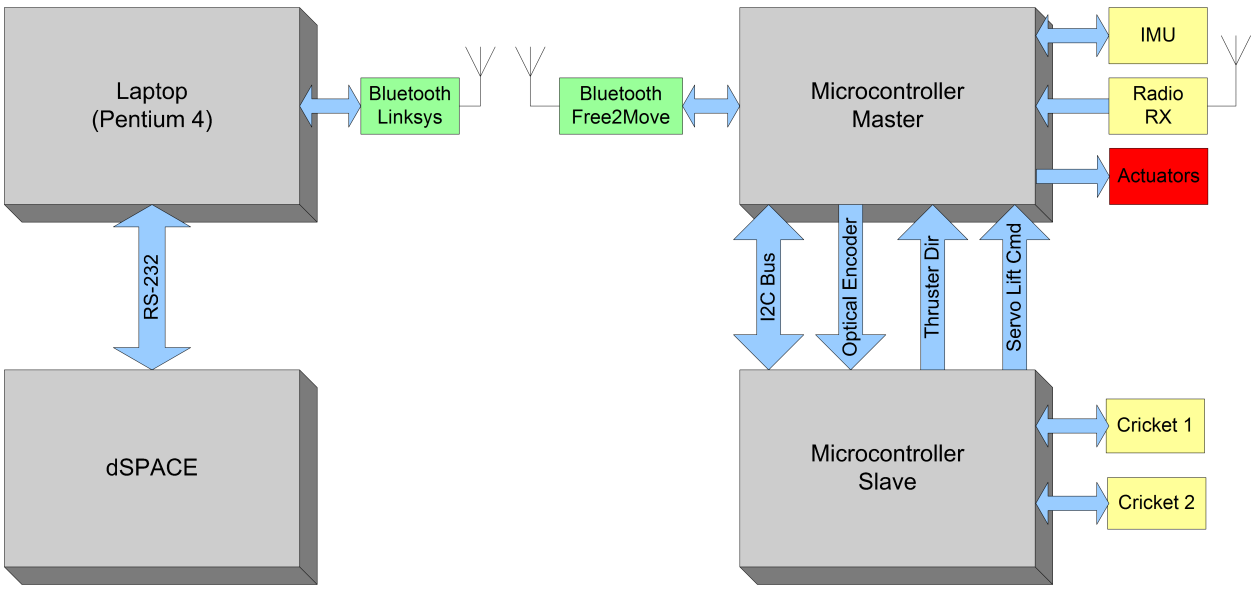


Figure 5.1: Block diagram of the autopilot top-level system architecture

at fixed time steps. This is invaluable, since it eliminates potential sources of error when debugging faulty algorithms.

The system was also used extensively to test the INS performance using a rotating platform. By creating a custom rotating platform controller block in Simulink, the computer was able to run all of the tests autonomously while collecting performance data.

5.1.1 Autopilot Implementation on dSPACE

We used a dSPACE 1103 system featuring a 333 MHz PowerPC. The system provides 16 multiplexed channels of 16-bit A/D, 8 channels of 16-bit D/A, 32 parallel channels of digital I/O, a digital and analog incremental encoder interface, hardware interrupts, a UART, a CAN bus interface, and a 20 MHz Texas Instruments DSP slave. The PowerPC processor and the supporting I/O electronics reside on a large

card with an ISA bus interface. We used the only ISA-equipped computer in our lab to host the dSPACE system. This 400 MHz PC was quite slow, but still managed to run dSPACE without any problems.

The hovercraft autopilot was implemented in Simulink using a combination of Matlab provided library blocks and custom S-functions (discussed in the next section). The model was compiled to C code for execution on the dSPACE system using the Real-Time Workshop. On dSPACE, the autopilot was executed in real-time using a fixed execution interval of 1 ms with overrun detection enabled. The Simulink autopilot block diagram appears below in Figure 5.2. We also created a visual pilot console to interface with the dSPACE system and control the operation of the autopilot. Appendix B contains a screenshot of the pilot console.

5.1.2 S-Functions

S-functions are user-defined functions that provide custom functionality within the Matlab environment. They are often used to encapsulate complex algorithms or leverage existing computer code in Simulink models. S-functions interface with Matlab's differential equation solvers and can handle continuous, discrete, and hybrid dynamical models. S-functions can also be compiled to object code using the Real-Time Workshop for execution on dSPACE or an alternate real-time target.

The hovercraft autopilot relies extensively on custom S-functions to perform the bulk of the computations. For example, there are several S-functions that manage communications with the hovercraft microcontrollers. These S-functions decode

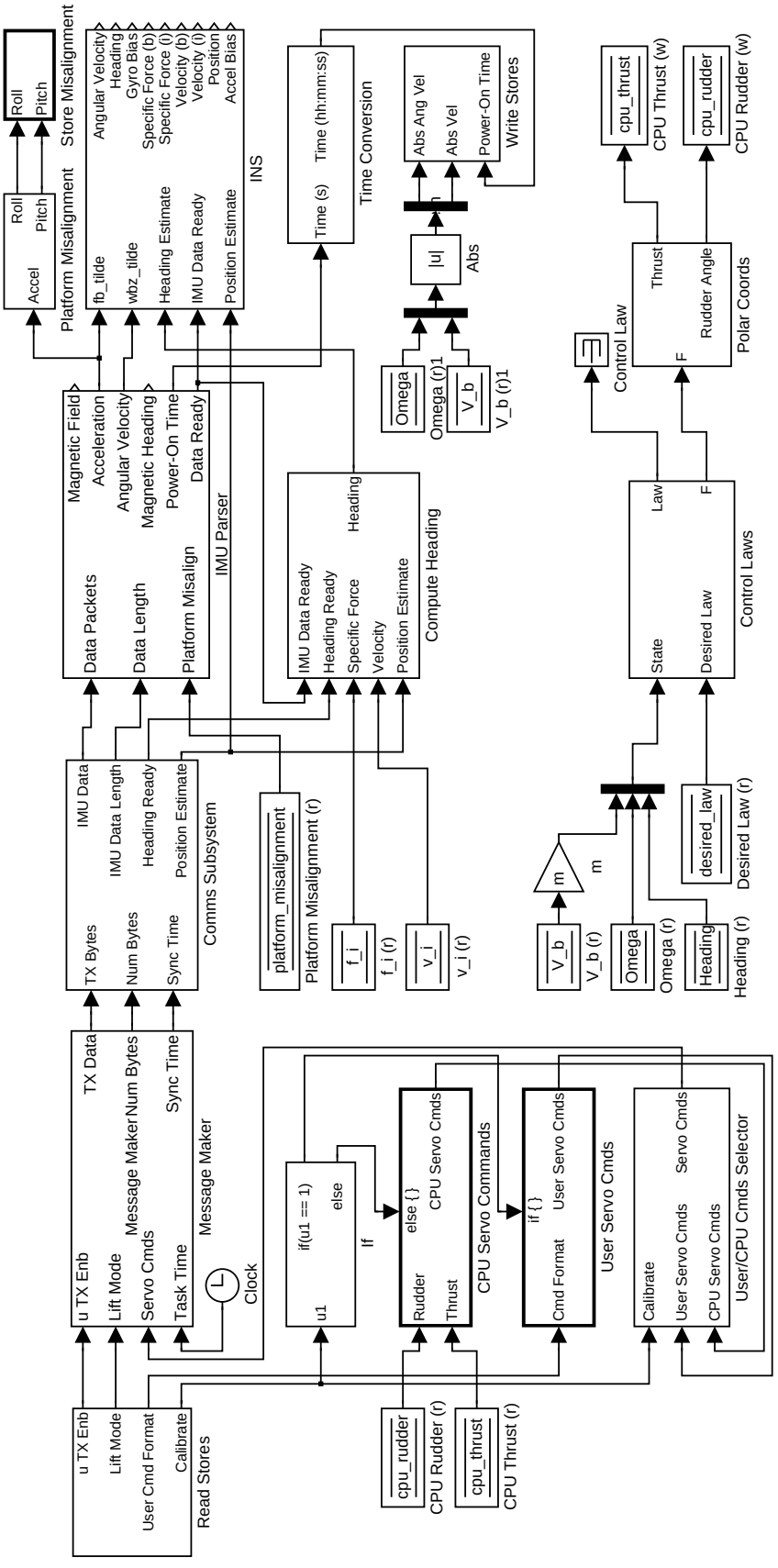


Figure 5.2: Simulink implementation of the hovercraft autopilot

messages from the hovercraft and prepare messages for uplink to the vehicle. Additionally, there are S-functions that implement the aided INS and parse IMU sensor messages. Each of the S-functions is written in ANSI C for computational efficiency and ease of integration with dSPACE.

5.2 Microcontrollers

The hovercraft features two PIC 18F8720 8-bit microcontrollers to handle low-level interfacing with the sensors and actuators. These microcontrollers are low-power processors running at 20 MHz with 128K of program flash memory and 3840 bytes of RAM. The 18F8720 microcontrollers provide five independent input capture/output compare channels, two dedicated 8-bit counters, two dedicated 16-bit counters, two hardware UARTs providing RS-232 and RS-485 functionality, a synchronous serial port module with I2C functionality, and 16 multiplexed channels of 10-bit A/D. In addition, high and low priority interrupts are supported.

We should note that while a full-fledged Pentium microprocessor could be used to execute the navigation and control algorithms onboard the hovercraft, the microcontrollers still provide valuable functionality. As lightweight systems rich in I/O, they are ideally suited to interfacing with sensors and performing time-critical tasks such as pulse width modulation (PWM) generation and decoding. Thus, their presence adds modularity to the system design.

We purchased two evaluation boards (TQFP 64/80) from Microchip for $49.00 each, with the 18F8720 microcontroller provided as a surface mount component.

The evaluation boards greatly simplified system development by fanning out the microcontroller pins to solderable pads and providing essential ancillary hardware including an RS-232 transceiver, 20 MHz crystal oscillator, and voltage regulator. The boards also provided an ICD2 port for in-circuit programming and debugging, 8 LEDs for visual feedback, and a small area for circuit prototyping.

The hovercraft microcontroller architecture uses a master-slave relationship. The master microcontroller manages all communications with the dSPACE system over the Bluetooth link. A small Bluetooth to RS-232 converter (provided by Free2Move) plugs into the serial port on the microcontroller board and provides the wireless connectivity. In addition, the master interfaces with the IMU via RS-232, decodes safety-pilot commands output in PWM format from the radio receiver, generates PWM control signals for the actuators, and runs the thrust fan RPM regulator.

The slave microcontroller controls the lift fan and interfaces with the two on-board Cricket units via RS-232. The two processors communicate with each other over the I2C bus. The slave microcontroller is actually the I2C bus *master* and controls when the master device gets to transmit. The reason is that there is a greater flow of time-sensitive information from the slave to the master microcontroller. Thus, the slave unit should have the ability to access the bus whenever it has information to transmit. The master microcontroller, on the other hand, is allowed to transmit up to 10 bytes of data every 10 ms.

The microcontrollers communicate with each other and the dSPACE system using messages. These messages contain minimal overhead and consist of a single-

byte header, message identifier, an optional message length, and the message data. There is no restriction on message length, although smaller messages are preferred in order to minimize Bluetooth link latency.[1] Refer to Appendix A for schematic drawings of the microcontroller system and custom supporting electronics.

5.3 Bluetooth

5.3.1 Introduction

Bluetooth is a wireless communications technology developed by Ericsson Mobile Communications in 1994. Although its original purpose was for cable replacement, many different types of devices now incorporate Bluetooth transceivers for generic wireless connectivity.

Bluetooth operates in the 2.45 GHz ISM band and employs frequency hopping for improved noise immunity. Seventy nine frequency channels, each separated by 1 MHz, are used in the hop sequence. Each frequency slot lasts for 625 μs and the hop pattern follows a pseudorandom sequence with a period of approximately 23 hours.

One of the key features of Bluetooth is the ability for devices to form ad-hoc networks easily on their own. Small networks called *piconets* are comprised of up to seven connected devices. These piconets can join together to form larger *scatternets* of interconnected devices.

[1] See section 5.3 for information on Bluetooth.

Bluetooth devices connected in a piconet form a star topology in which one or more slaves communicate exclusively with a single master. Point to point and point to multipoint communications are supported between master and slaves, but direct communications between slaves are not. The master device transmits in even numbered time slots and the slaves transmit in odd numbered slots. A slave must be *polled* by the master in order to transmit in the following time slot. In this time division duplex scheme, full duplex communications are achieved between the master and slave devices.

Bluetooth devices may also dynamically switch roles to participate in different piconets. For example, a master of one piconet may be a slave in another piconet. The ability to form scatternets is important since Bluetooth devices are limited in range. Class 2 devices support a 10 m range while Class 1 devices increase their output power and range to about 100 m. Scatternets allow devices separated by large distances to communicate with each another.

The Bluetooth specification was also designed with battery conservation in mind. There are three low-power modes that provide various degrees of power savings. These are, in order of increased power savings: sniff mode, hold mode, and park mode. In sniff mode, the duty cycle of the receiver is reduced, while in park mode, the device does not participate at all in the piconet. The clock continues to run, however, so that the device remains synchronized with the master.

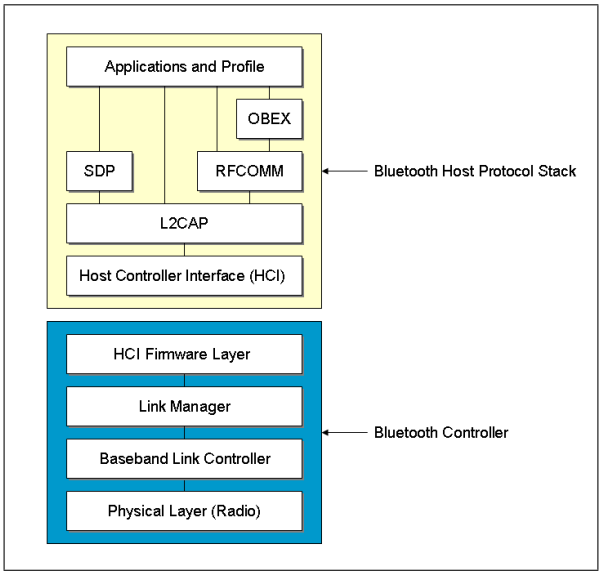


Figure 5.3: The Bluetooth protocol stack

5.3.2 The Bluetooth Stack

Bluetooth features a stack-based architecture similar to the OSI model in which higher stack layers rely on functionality provided by the lower layers. A diagram of the Bluetooth architecture is shown in Figure 5.3. The *Physical Layer* is the lowest stack layer and provides the basic RF functionality. The physical layer is implemented in hardware as an ASIC (Application Specific Integrated Circuit).

The next layer is the *Baseband Link Controller Layer*. This layer packages raw data into one of thirteen standard Bluetooth packets, performs channel encoding and decoding, performs CRC generation and checking, and handles the *Automatic Repeat Request* (ARQ) protocol. There are different packets defined to carry voice, data, or a combination of both. Depending on the required data rate, these packets may occupy one, three, or five consecutive $625\mu s$ time slots. A single slot carries up to 27 bytes and 5 slots can transport 339 bytes, yielding symmetric data rates of 172.8

kb/s and 433.9 kb/s respectively. Asymmetric data rates can also be achieved by using different packet types for the upstream and downstream directions. Observe that data can be packed more efficiently into multiple time slots since the radios do not retune their frequency synthesizers during this time.

Bluetooth provides support for optional bit error detection and correction at the baseband layer. If enabled, the ARQ protocol ensures that all packets are eventually received free from bit errors. When a received packet fails a CRC check, the receiver may request a packet retransmission. Packet retransmission continues until the packet is received correctly or a timeout occurs. Bluetooth also features two forward error correction (FEC) schemes. In 1/3 FEC, each bit is repeated 3 times, resulting in a 2/3 decrease in available bandwidth. Alternatively, the 2/3 FEC is a (15,10) shortened Hamming code which produces one extra bit for every two bits processed. The choice to use ARQ or FEC is dependent on the type of data link used.

As mentioned earlier, Bluetooth supports the transmission of voice and data. For voice data, synchronous connection-oriented (SCO) links are used with FEC enabled and ARQ disabled. SCO channels provide the lowest possible latency since the transmission slots and payload sizes are predefined. On the other hand, raw data packets are transmitted using asynchronous connection-oriented (ACL) links. There are 7 different ACL packets providing payload capacities ranging between 17 and 339 bytes with optional 2/3 FEC. All packets must contain a CRC checksum since the ARQ protocol is required for ACL links.

Moving up the stack, the next layer is the *Link Manager Layer*. This layer implements the Link Manager Protocol and handles establishing and maintaining physical connections with devices, configuring link parameters, and exchanging security-related messages. There are special single-slot packets called *protocol data units* (PDUs) for accomplishing these tasks. The *HCI Firmware Layer* sits on top of the Link Manager Layer and provides a common interface to the lower layer functionality.

The Physical, Baseband Link Controller, Link Manager, and HCI Firmware layers provide low-level Bluetooth functionality and collectively form the *Bluetooth controller*. Typically, the Bluetooth controller is implemented in hardware, and may even be a single integrated circuit. A Bluetooth controller is incorporated into a *host device* to provide Bluetooth connectivity.

The next group of layers constitute the Bluetooth Host Protocol Stack. The functionality specified by these layers is provided by software which may be executed on the Bluetooth host or on a separate processor. To achieve separation between the high level protocol processing and the low-level controller tasks, the Bluetooth specification provides an optional *Host Controller Interface (HCI) Layer*. The HCI provides a common interface to the Bluetooth controller and supports host configuration, link control, and baseband commands. The choice to support this layer depends on the particular type of Bluetooth device and whether there is a need for modularity.

The *Logical Link Control and Adaptation Protocol (L2CAP) Layer* is the primary buffer between high level Bluetooth-independent applications and the Blue-

tooth controller. The L2CAP is responsible for providing multiple logical channels, multiplexing data from multiple services, and segmenting and reassembling datagrams. For example, the payloads carried by TCP packets are too large to fit inside any of the Bluetooth packets. The data must be segmented prior to transmission and then reassembled on the receiving side. The L2CAP also negotiates quality of service (QoS) parameters with other devices and tries to ensure that performance expectations are met.

At this point, the Bluetooth protocol stack splits in different directions. The Service Discovery Protocol (SDP) and RFCOMM sit on top of the L2CAP layer. The SDP enables devices to query each other for information about supported services. RFCOMM is a protocol that emulates serial port communications over a Bluetooth link. The protocol provides support for up to 60 simultaneous serial port connections. Other high-level applications may interface directly with L2CAP to send custom-defined data to connected devices.

5.3.3 Bluetooth in Control Systems

Inexpensive wireless technology such as Bluetooth has paved the way for new distributed control system architectures. In a distributed control system, the controller is physically separated from either the sensing node, the actuation node, or both nodes. These systems arise naturally when there is a desire to control one or more physically separated nodes with a central controller.

Like any wireless communications technology, Bluetooth presents several challenges when used in distributed control systems. First, despite fast frequency hopping, Bluetooth data packets can be corrupted by channel noise. Packet retransmission is viable only if the controller sample rate is slow enough. For systems with fast dynamics, lost sensor or actuator data can cause the system to become unstable.

Second, Bluetooth data links exhibit time-varying latencies. These delays are caused by protocol overhead, master/slave polling, and available network bandwidth. They are more pronounced in ACL data links since slaves do not have reserved transmission slots and payload sizes are dynamic. In addition, data delays may result from inefficient data packing implementations at the higher application layers. For example, we have observed that the RFCOMM protocol for serial port emulation buffers several bytes of serial data before bursting it across the data link.

Packet delays in a distributed control system have a destabilizing effect on the system dynamics. For example, it is well known that a linear system can tolerate a maximum constant loop delay given by $\Delta_{max} = \phi_m/\omega_c$ where ϕ_m is the phase margin at the critical frequency, ω_c. It is more difficult to analyze the effect of time-varying delays on system stability. If, for example, the delay is known to lie in the interval $[\Delta_{min}, \Delta_{max}]$, the controller can be designed to handle a representative value of the delay, Δ. Common choices include $\Delta = \Delta_{min}$, $\Delta = \Delta_{max}$, and $\Delta = \Delta_{avg}$. The problem is that the feedback controller can be stable for constant delays equal to Δ_{min} and Δ_{max} and fail to be stable when the delay is varying [25].

The Jitter Margin Theorem [25] gives a sufficient condition for linear system stability when the time-varying delay lies in the interval $[0, Nh]$ where h is the

sample time and N is a real number. For nonlinear system dynamics, the analysis is significantly more complex.

For optimal control system performance, lost packets and packet delay must be dealt with carefully. For linear systems, there are precise methods for handling lost and delayed packets, depending on whether the data was sent from a sensor or sent to an actuator. For a constant sensor delay that is less than one sample period, the optimal control is given by:

$$u(k) = -L \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \tag{5.1}$$

where $x(k)$ is the state vector, $u(k-1)$ is the previous control, and L is a constant feedback gain vector. For time-varying delays, there are two options. Equation (5.1) may be used with the mean delay, but this yields suboptimal results. Alternatively, the vector L may be replaced with a time-varying feedback gain vector, $L(\tau_k^{sc})$, where τ_k^{sc} is an estimate of the sensor-to-controller delay [25].

Packet loss due to bit errors poses additional challenges for achieving robust system performance. As an example, consider the linear dynamics:

$$\begin{aligned} x(k+1) &= ax(k) + u(k) + d(k), \quad a > 0 \\ y(k) &= x(k) \end{aligned} \tag{5.2}$$

where $d(k)$ is a white noise disturbance process with unit variance. The optimal control for the cost function $J = \mathrm{E}\{x^2\}$ is given by:

$$u(k) = -a\hat{x} \tag{5.3}$$

where $\hat{x}$ is the best estimate of the state at time k. For a lost packet recovery strategy in which retransmission is attempted only once, it may be shown that it is impossible to stabilize the system if $|a|q > 1$, where q is the probability of receiving a corrupted packet [25].

The techniques described above for mitigating packet delay and loss are applicable to linear system dynamics only. Nonlinear plant dynamics complicate matters substantially and require specialized analysis that depends on the type of nonlinearities present. In many cases, simulation is the only tool available for quantifying the effect of packet delay and loss on system performance and stability.

5.3.4 Autopilot Delays

There are three main sources of delay present in the hovercraft autopilot. These delays are time-varying and include the sensor-to-controller delay, τ^{sc}, the controller-to-actuator delay, τ^{ca}, and an actuator delay, τ^{a}. The controller delay is negligible and limited to a maximum of 1 ms by the dSPACE RTOS. Of course, there is always the potential for dropped packets in the system.

Through experimentation, we have shown that the two Bluetooth delays, τ^{sc} and τ^{ca}, are primarily dependent on the particular implementation of the RFCOMM protocol in the Bluetooth stack. Recall that RFCOMM is a protocol for serial port emulation over Bluetooth. In addition, these delays are heavily influenced by the data rates required in the upstream and downstream directions. Data that we collected indicates that higher data rates incur more delay.

The actuator delay, τ^a, is dependent on the arrival time of the control commands with respect to the servo update time. The servos and speed controllers update their outputs every 20 ms. Thus, τ^a lies in the interval $[0, .020)$ and is time-varying.

5.3.5 Autopilot Bluetooth Devices

We considered several options for connecting the hovercraft to a Bluetooth network. The best option in terms of size, weight, power, and ease of use was a Bluetooth serial port plug manufactured by Free2Move ($113). This device is a self-contained Bluetooth host which implements the RFCOMM protocol in firmware and provides wireless serial port cable replacement. There are no drivers to install and the device plugs directly into a standard RS-232 serial port. The Free2Move serial port plug is a Class 1 Bluetooth device with a range of 100 m in open air.

We purchased two of these units to provide Bluetooth connectivity on the hovercraft and dSPACE system. A device like the Free2Move is ideally suited for dSPACE, which does not have an open operating system on which to compile custom drivers. We also purchased a USB Bluetooth dongle manufactured by Linksys (USBBT100) to compare performance with the Free2Move device. The Linksys dongle is a Bluetooth controller only and requires a PC to provide the additional functionality specified by the Bluetooth Host Protocol Stack. We used the open source "BlueZ" Bluetooth protocol stack integrated in the Linux 2.6 kernel. BlueZ

is a multithreaded and stable implementation of the Bluetooth stack which achieved qualification as a Bluetooth subsystem in April 2005.

5.3.6 Experimental Performance

We created a simple experiment to quantify the delays present in the Bluetooth network. Specifically, we added custom software and hardware support to measure the uplink (controller to hovercraft) and downlink (hovercraft to controller) delays precisely. We then quantified the wireless link performance for different pairings of devices and for various data rates.

Synchronization between the dSPACE system and the microcontroller was achieved with a single wired connection. The dSPACE system generated a 1 kHz timing reference and a strobe to mark the transmission of a special *CLOCK_TIME* message. This packet contained only the current dSPACE system time, truncated to 4 bytes.

The strobe signal generated a microcontroller interrupt which caused a free-running delay timer to be reset. When the *CLOCK_TIME* message was finally received by the microcontroller over the Bluetooth link, the timer value was read and appended to the message. The *CLOCK_TIME* message was then immediately transmitted back to the dSPACE system. Using the original timer value contained in the message and the uplink delay estimate provided by the microcontroller, the dSPACE controller was able to compute the downlink delay.

We tested a variety of device pairings and data rates to determine the lowest achievable network latencies. The best results were obtained with the hovercraft Free2Move device configured as the Bluetooth master. A Pentium 4 laptop running the Linux 2.6 kernel with BlueZ support was used for latency tests between the Linksys USB Bluetooth dongle and the Free2Move device. The Linux 'rfcomm' utility provided RFCOMM protocol support for serial port emulation over Bluetooth. Additionally, a small application running on the laptop relayed data between the PC and the dSPACE system over an RS-232 serial cable.

All serial devices were set to 115.2 kbps for minimal latency. The Free2Move devices feature a software utility to configure network settings and operating parameters. We experimented with a variety of different settings and evaluated their impact on network latency. Many of the settings had little or no effect on network performance. We observed, however, that minimal latency was achieved by selecting the 'optimize for latency' setting and disabling the 'quality of service' option.

The data from our network latency tests appears below in Table 5.1. Surprisingly, the lowest latency was achieved for connections between the Free2Move and Linksys devices. For uplink/downlink data rates of 2.9/22.0 kbps, the average round trip delay was 77.54 ms. On the other hand, two paired Free2Move units also transmitting at 2.9/22.0 kbps yielded a round trip latency of 162.98 ms, which is twice as large! We attribute the poor performance to a suboptimal implementation of the Bluetooth stack in the Free2Move devices. In contrast, the BlueZ stack is a highly optimized implementation of the protocol running on a fast processor.

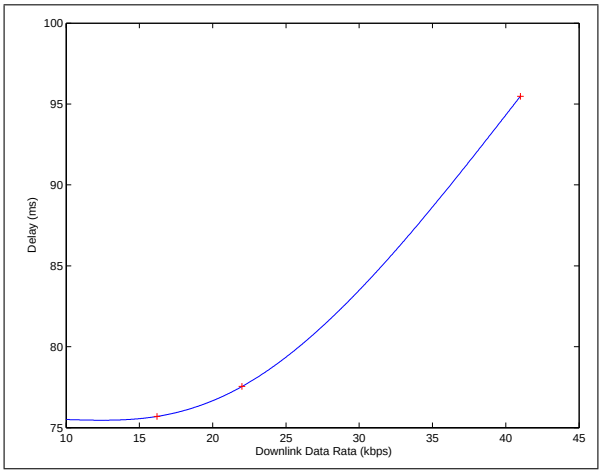


Figure 5.4: Bluetooth round trip delay vs downlink (hovercraft to controller) data rate

We see from the data that the best performance was obtained for lower data rates. The data rates required by the autopilot in the uplink and downlink directions are highly asymmetric. A much larger downlink data rate is needed to service the high-bandwidth traffic generated by the IMU sensor. A graph of the total round trip delay versus downlink data rate is shown in Figure 5.4. The graph indicates that a large reduction in latency is obtained by halving the downlink data rate from 41.0 to 22.0 kbps. Beyond this point, further decreases in the data rate yield only meager improvements in latency. During the test, the downlink data rate was varied by selectively discarding IMU packets. Thus, an important engineering tradeoff exists between the effective IMU sample rate and the received data latency.

	Mean	Std Dev	Min	Max
Free2Move/Linksys (2.9/41.0) kbps				
Uplink	51.04	15.09	22	118
Downlink	39.02	9.23	20	74
Round trip	95.48	18.01	58	157
Free2Move/Free2Move (2.9/41.0) kbps				
Uplink	66.01	10.22	40	102
Downlink	95.36	12.02	59	141
Round trip	160.21	13.86	124	194
Free2Move/Linksys (2.9/22.0) kbps				
Uplink	40.81	10.61	22	90
Downlink	34.32	8.18	19	67
Round trip	77.54	14.44	46	110
Free2Move/Free2Move (2.9/22.0) kbps				
Uplink	62.76	11.13	40	106
Downlink	100.01	13.64	53	161
Round trip	162.98	16.58	113	208
Free2Move/Linksys (2.9/16.2) kbps				
Uplink	38.83	8.80	20	68
Downlink	32.62	8.49	14	65
Round trip	75.69	13.82	49	126

Table 5.1: Bluetooth network latency data. (All times are in ms.)

5.4 Inertial Navigation System (INS)

In this section, we describe the physical implementation of the two-dimensional aided INS. The reader should refer to section 6.2.1 for information regarding the INS performance and testing procedure.

5.4.1 Inertial Measurement Unit (IMU)

We used a Microstrain 3DM-GX1^{TM}IMU that features three orthogonal MEMs accelerometers, gyroscopes, and magnetometers packaged in a small (2.5 by 3.5 inch) case. The accelerometers are manufactured by Analog Devices (part number ADXL103) and have a full-scale range of $\pm$1.7g. The gyroscopes are also manufactured by Analog Devices (part number ADXRS150) and measure angular velocities in the range of $\pm 300^{\circ}/s$.

A microcontroller is integrated with the inertial sensors and performs physical units scaling, axis misalignment correction, and temperature compensation. Microstrain performs a full system calibration of each unit and stores the compensation parameters in the microcontroller's flash memory. In addition, the microcontroller optionally runs a proprietary filtering algorithm called Fusion that corrects the gyroscope biases and outputs a gyro-stabilized 3-D orientation matrix.

The main problem with Fusion is that it performs abysmally when the IMU is subjected to prolonged angular rotation or linear acceleration. Additionally, the Fusion algorithm fails to cope adequately with transient angular velocities that approach the gyroscope's limits. When either of these conditions occurs, the reported orientation drifts wildly and is unusable. The user must then bring the IMU to rest and manually resample the gyro biases (a process that takes several seconds) in order to reset the filter. The shortcomings of the Fusion algorithm prompted our development of a two-dimensional INS.

The basic execution interval on the IMU is a *tick* with a default value of 6.5536 milliseconds. The message update rate varies from 1 to 3 ticks depending on the particular data message requested and whether filtering and gyro bias compensation are enabled. Raw sensor measurements require the least amount of processing time and are transmitted once per tick. Data is transmitted serially using the RS-232 or RS-422 electrical protocol.

5.4.2 Aiding Sensors

The primary INS aiding sensor on the hovercraft is a "Cricket" positioning device.[2] The Cricket system uses a combination of ultrasound and RF to estimate position in two dimensions with respect to a user-defined coordinate system. It is essentially an indoor GPS replacement.

Although we originally anticipated using the magnetometer to aid the heading filter, we encountered significant problems due to the presence of steel beams in the floor and nearby time-varying magnetic fields produced by the electric motors. Fortunately, we were able to use two Cricket units to provide reasonable heading estimates.

5.4.3 Real-Time Processor

The INS processor is a 333 MHz PowerPC embedded in the dSPACE system. dSPACE executes the INS code in parallel with the hovercraft controller using a

[2]See section 5.5 for a description of the Cricket system.

fixed time step of 1 ms. All calculations must complete within the allotted time, or an overrun will be signalled and execution will halt.

5.4.4 Software

The INS is implemented in C code using the Matlab S-function framework.[3] We used the free "meschach" [26] matrix library written in ANSI C to handle the matrix computations inherent to Kalman Filtering. The meschach library is small, fast, and portable since it adheres to the ANSI C standard. Portability ensures that the library works properly on both the PC (Intel) and dSPACE (PPC) hardware platforms.

5.4.5 Discrete Error Model

The noise covariance matrices, Q_i and measurement covariance matrices R_i, $(i = 1, 2)$ were determined experimentally by collecting sensor data for one hour with the sensor stationary. Appropriate noise powers for the random walk processes were selected using simulation. Software functionality was provided to permit adjustment of the noise parameters in real-time for INS performance tuning. The nominal covariance matrices used in the Kalman Filter equations are:

$$Q_1 = \begin{bmatrix} \text{4.9e-4} & 0 \\ 0 & \text{5e-4} \end{bmatrix}, \quad R_1 = [\text{1.6e-4}] \tag{5.4}$$

[3] See section 5.1 on Simulink and dSPACE.

for the heading filter and

$$Q_2 = \begin{bmatrix} 3.4\text{e-}2 & 0 & 0 & 0 \\ 0 & 2.7\text{e-}2 & 0 & 0 \\ 0 & 0 & 5\text{e-}4 & 0 \\ 0 & 0 & 0 & 5\text{e-}4 \end{bmatrix}, \quad R_2 = \begin{bmatrix} 5.6\text{e-}4 & 0 \\ 0 & 5.6\text{e-}4 \end{bmatrix} \tag{5.5}$$

for the velocity and position filter.

Equations (3.31) and (3.32) were discretized using a fixed update rate of 6.5536 ms. The resulting discrete error dynamics model is given by:

$$\begin{aligned} \delta\mathbf{x_1}(t_k) &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \delta\mathbf{x_1}(t_{k-1}) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{w_1}(t_{k-1}), \\ \delta\mathbf{x_1} &= [\delta\theta \;\; \mathsf{b_g}]^{\mathrm{T}} \end{aligned}$$

and

$$\begin{aligned} \delta\mathbf{x_2}(t_k) &= \Phi_2(t_k, t_{k-1})\delta\mathbf{x_2}(t_{k-1}) + G_2(t_{k-1})\mathbf{w_2}(t_{k-1}), \\ \delta\mathbf{x_2} &= [\delta\mathsf{v_x} \;\; \delta\mathsf{v_y} \;\; \delta\mathsf{r_x} \;\; \delta\mathsf{r_y} \;\; \mathsf{b_{a_x}} \;\; \mathsf{b_{a_y}}]^{\mathrm{T}} \end{aligned}$$

where

$$\Phi_2(t_k, t_{k-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 & c_1\cos(\hat{\theta}(t_{k-1})) & -c_1\sin(\hat{\theta}(t_{k-1})) \\ 0 & 1 & 0 & 0 & c_1\sin(\hat{\theta}(t_{k-1})) & c_1\cos(\hat{\theta}(t_{k-1})) \\ c_1 & 0 & 1 & 0 & c_2\cos(\hat{\theta}(t_{k-1})) & -c_2\sin(\hat{\theta}(t_{k-1})) \\ 0 & c_1 & 0 & 1 & c_2\sin(\hat{\theta}(t_{k-1})) & c_2\cos(\hat{\theta}(t_{k-1})) \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G_2(t_{k-1}) = \begin{bmatrix} \cos(\hat{\theta}(t_{k-1})) & -\sin(\hat{\theta}(t_{k-1})) & 0 & 0 \\ \sin(\hat{\theta}(t_{k-1})) & \cos(\hat{\theta}(t_{k-1})) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$c_1 = \Delta t$$

$$c_2 = \frac{(\Delta t)^2}{2}$$

and Δt is the fixed IMU sample rate. The error state vectors, $\delta\mathbf{x_1}$ and $\delta\mathbf{x_2}$ are initially set to 0, and the noise covariance matrices, P_1 and P_2, take initial values:

$$P_1(t_0) = \begin{bmatrix} 1.6\text{e-}4 & 0 \\ 0 & 1 \end{bmatrix}$$

$$P_2(t_0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5.6\text{e-}4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5.6\text{e-}4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In order to maintain synchronization with the IMU, an 'IMU Data Ready' strobe tells the Kalman Filters when to propagate the error state. The synchronization signal is necessary because the Bluetooth transmission delay is variable.

The Kalman Filters incorporate new heading and position measurements when they become available and update the error state estimates accordingly. The outputs of the heading filter are the bias-corrected angular velocity, the corrected heading, and the gyro bias. Similarly, the outputs of the velocity/position Kalman Filter are the bias-corrected specific force vector, the corrected velocity and position in inertial coordinates, and the accelerometer biases.

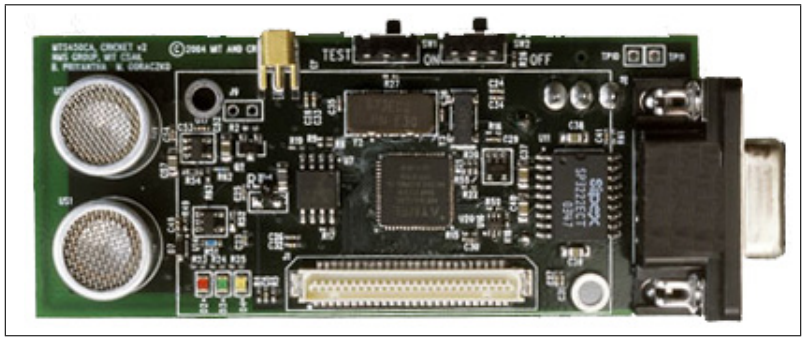


Figure 5.5: Cricket mote (Photo courtesy of MIT CSAIL)

5.5 Cricket Positioning System

Cricket is a low-cost, scalable, and robust indoor positioning system. The system uses a combination of ultrasound and RF to estimate position in two dimensions with respect to a user-defined coordinate system. Although overall positioning accuracy is dependent on the quality of the system calibration, accuracies on the order of a few centimeters may be easily achieved. Cricket also provides accurate dynamic positioning, as long as the sensor speed is well below the speed of sound (340.29 m/s).

Cricket was originally conceived and developed by Hari Balakrishnan and Nissanka Priyantha at the Massachusetts Institute of Technology [18, 19]. MIT maintains a relationship with Crossbow Technology to commercially produce the Cricket system. Crossbow manufactures the Cricket boards and distributes a set of eight sensors packaged with MIT's Cricket firmware. The Cricket unit, shown in Figure 5.5, is a modified Mica2 *mote* originally developed by the University of California, Berkeley.

The Cricket motes are small, low-power sensor platforms that feature an Atmel 8-bit microcontroller, an ISM-band radio transceiver manufactured by Chipcon, two ultrasonic transducers, and serial interface circuitry. The units run the TinyOS operating system and are programmed in NesC, a variant of ANSI C for embedded processors.

We have completely redesigned the Cricket firmware from the ground up in order to greatly improve positioning accuracy when the device is moving. In the following discussion, we describe our implementation of the Cricket system at the University of Maryland. Later, we will highlight the major differences between the two systems.

5.5.1 Cricket Entities and System Architecture

Cricket consists of two entities: roving clients and fixed beacons. Beacons play the role of satellites in a GPS system while clients access the system to obtain position estimates. Unlike GPS, Cricket uses the large difference in the propagation velocities of light and sound to compute ranges. Individual client-to-beacon ranges are computed by differencing the arrival times of an RF pulse and a 40 kHz ultrasonic chirp.

Multiple clients access the Cricket system using a time-division access protocol. Each client has a reserved 100 ms time slice in which to ping the beacons with an ultrasonic chirp and await responses. The maximum range of the transmitted ultrasound is approximately 30 feet, corresponding to about 30 ms of flight time.

Beacons within range of the client measure the time of flight of the ultrasonic chirp, determine the speed of sound using the local temperature, and compute the range estimate. The beacons then report the range back to the requesting client.

In order to reduce system complexity and network overhead, the beacons do not explicitly coordinate with each other when accessing the wireless channel. Instead, the beacons implement a carrier sense multiple access (CSMA) protocol with random backoff in order to mitigate RF packet collisions. Each beacon wishing to transmit range information initially delays a random number of bytes by uniformly sampling the interval $[1, 64]$. When the backoff condition is satisfied, the beacon determines whether the link is in use by polling the RSSI (Received Signal Strength Indicator). If the link is clear, the beacon switches to transmit mode and sends the message.

Observe that all entities in the network receive every radio transmission. Entities must determine if they are the intended message recipient by examining a specific message field. After pinging the beacons, the client awaits range estimates. If at least two ranges are received by the end of the time slice, the client can estimate its position in the plane.[4] Note, however, that the computed position is ambiguous. A third range measurement helps to resolve the positioning ambiguity, and additional range measurements, if available, improve the position estimate in a least-squares sense.

[4]The range message contains the ceiling height and the client knows its height above the ground.

5.5.2 Time Synchronization

Time synchronization is vital to the Cricket system for two reasons. First, clients must remain synchronized with each other so that they respect the time division multiple access protocol. Second, beacon and client timers must be synchronized in order to measure the ultrasound time of flight.

Time synchronization among clients is controlled by the *master* client. Only one client in the network may be designated the master. Each second, the master device issues an RF synchronization message which causes the remaining clients to restart their internal timers. The Cricket boards feature a highly accurate 32.768 kHz oscillator for precise timing. This is significant because it permits upgrading the microcontroller oscillator without having to modify the low-level timebase-dependent assembly code.

An RF message is also used to synchronize the beacon and client timers when the client issues the ultrasonic chirp. After accounting for fixed processing delay, the clocks are synchronized to within 1 μs.

5.5.3 Position Determination

The client computes its position by solving a system of nonlinear equations. At least two range measurements are required to estimate position in the plane. With two measurements, position determination amounts to solving the intersection of two circles in the plane. The problem is that the circles may intersect in two points.

The ambiguity may often be resolved by selecting the position solution that is closest to the previous estimate.

Additional measurements, if available, can be used to increase the accuracy of the estimate and eliminate the solution ambiguity. Following the GPS model, we assume that the range equations take the following form:[5]

$$\rho_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + h^2} + b, \quad i = 1, 2, \ldots n \tag{5.6}$$

where (x_i, y_i) denotes the location of the ith beacon, h is the fixed vertical height between the client and the beacons, and b is an error term included to make the system of equations consistent. In a GPS system, b is called the 'clock bias' and accounts for the offset between the satellite and receiver clocks. Since the speed of light is constant, b represents a distance error caused by the clock offset.

In the Cricket system, the meaning of b in (5.6) is slightly different. First, the speed of sound in ambient air is not constant. Although we use a highly nonlinear model to compute the speed of sound based on local temperature and humidity [27], there will inevitably be small errors due to variable temperature gradients in the propagation path. The other difference is that the clock offset in the Cricket system is minimal since the clocks are synchronized to within 1 μs. These two observations imply that b should be thought of as a small error term to make the system of range equations consistent. In fact, the magnitude of b provides information about the quality of the position estimate.

[5]The GPS range equations replace h with $(z - z_i)$, since altitude is variable. Thus, four range measurements are needed to solve position in three dimensions.

Fortunately, there is an elegant and computationally efficient solution to the range equations attributable to S. Bancroft [28]. Using some clever algebraic manipulations, the Bancroft algorithm provides a least-squares solution to the nonlinear equations and requires solving only a linear system and the roots of a quadratic equation. Three range measurements are required to determine the three unknowns in (5.6). Additional measurements are easily incorporated by the Bancroft algorithm and used to refine the position estimate in a least-squares sense.

5.5.4 Cone Angle Errors

A non-negligible source of error in the Cricket system results from nonlinearities in the ultrasonic transducers. The transducers are physical devices that convert electrical signals to ultrasound and vice-versa. As non-ideal devices, they exhibit energy conversion nonlinearities that are dependent on the angle of the transmitted and received ultrasound.

Figure 5.6 shows a diagram of a Cricket unit with a cone drawn to illustrate the spatial limitations of the transducer. We have observed that the transducer efficiency rapidly decreases as the cone angle, θ increases. In fact, at an angle of about 45°, the Cricket units are completely unable to detect the transmitted ultrasound. Depending on ceiling height, the ultrasound cone angle can severely limit the area covered by each beacon.

While the coverage problem may be solved by increasing the number of units, there is a more subtle issue caused by the cone angle effect. The problem is that

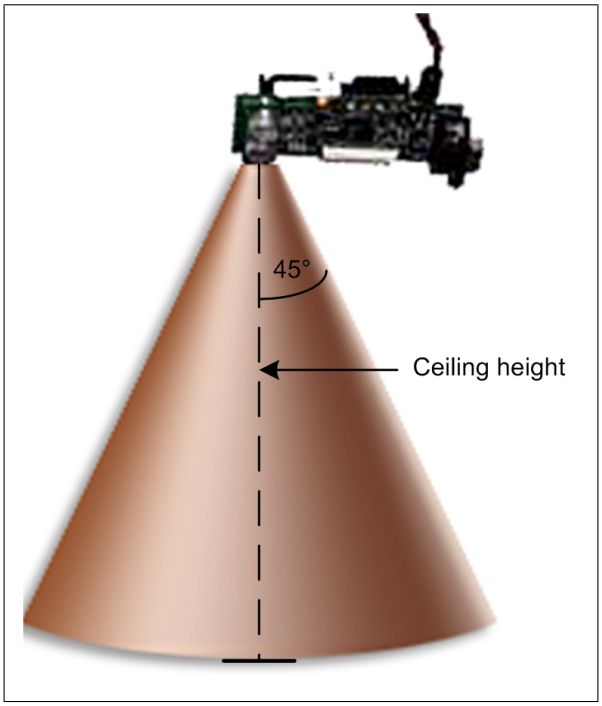


Figure 5.6: Illustration of the Cricket ultrasonic cone. Figure is not to scale.

the variability in sound energy produces errors in the range measurements that are dependent on the cone angle. The amount of energy present in the received ultrasound determines how long it takes the output from the envelope detector circuit to cross a fixed threshold.

Fortunately, the range measurements may be compensated for cone angle effects in the two-dimensional setting (i.e., if the client-to-beacon height is known). We devised an experiment to quantify the effect of cone angle on range error. A preliminary step in the experiment was to select a ceiling-mounted beacon and use a plumb line to project the location of the ultrasonic transducer onto the floor. We marked the location and then measured the vertical height, h, precisely. To achieve a

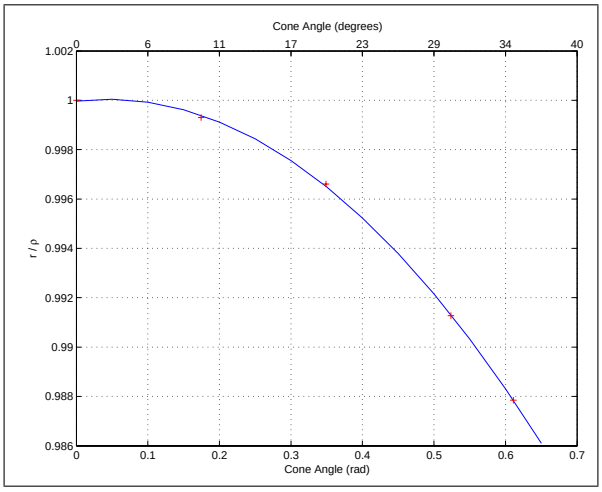


Figure 5.7: Ratio of true range to measured range as a function of cone angle

desired cone angle, θ, the client was positioned on a circle of radius $h\tan(\theta)$, centered at the mark. The true range at a particular cone angle was given by $r = h/\cos(\theta)$.

For each cone angle, we collected 240 measurements, ρ, (corresponding to 4 minutes of data) and averaged them. We then computed the error ratio $e = r/\rho$ for each cone angle. The data appears below in Table 5.2 and is plotted in Figure 5.7. In addition, we fit the data to a second-order polynomial with equation:

$$e(\theta) = (-0.038)\theta^2 + (.0032)\theta + 1 \tag{5.7}$$

The correlation coefficient was 0.9999.

Cone Angle	**True Range** (r)	**Measured Range** (ρ)	**Ratio** r/ρ
0°	270.48 cm	270.48 cm	1
10°	274.65 cm	274.83 cm	0.99935
20°	287.84 cm	288.83 cm	0.99657
30°	312.32 cm	315.07 cm	0.99127
35°	330.20 cm	334.25 cm	0.98788

Table 5.2: Range data for various cone angles and a ceiling height of 270.48 cm

Using error model (5.7), a measured range can be compensated for cone angle errors by solving:

$$r(\theta) = e(\theta)\rho \tag{5.8}$$

The problem is that we do not know the value of θ directly. However, by substituting for r in equation (5.8), we obtain:

$$\begin{aligned} \frac{h}{\cos(\theta)} &= e(\theta)\rho \\ e(\theta)\cos(\theta) - \frac{h}{\rho} &= 0 \end{aligned} \tag{5.9}$$

Thus, we must solve this nonlinear equation for θ and back-substitute to compute $r(\theta)$.

We implemented Newton's method on the client device to solve equation (5.9) and compensate the range measurements. For a tolerance of 1e-6, the solution typically converges within 5 iterations. Depending on cone angle, the improvement in range accuracy can be significant (several centimeters). The result is a lower positioning variance since the range errors are *normalized* and not dependent on the geometry of the active beacons with respect to the client.

5.5.5 Performance

Positioning accuracy depends on several factors including the quality of the system calibration, whether the client is static or moving, whether cone angle compensation is enabled, and the accuracy of the computed speed of sound. Performance is most limited by the system calibration and how accurately the measured beacon positions reflect the true sensor locations. Positioning accuracy of ± 10 cm

(client speed <4 m/s) is achievable using only a plumb line and a measuring tape to calibrate the system.

The calibration task is simplified if the Cricket network is confined to a single room featuring a drop ceiling with fixed-size panels. These panels provide a clear visual reference for defining a coordinate system. In future work, we would like to implement an autonomous calibration solution using a wheeled-robot with odometry, aided by range measurements. Such a system would provide a means for calibrating a disjoint network spanning moderate to large distances where the use of a measuring tape would be impractical.

Figure 5.8 depicts the static performance of the Cricket system. Data was collected for two minutes with the client stationary. The circular region denotes the area in which 50% of the measured ranges fall. This region represents the circular error probability (CEP) and has a radius of .52 cm.

5.5.6 MIT System Differences

There are several notable differences between our system and the original MIT system. The primary difference is that MIT system reverses the roles of the clients and beacons. In the MIT system, the clients are listeners and receive ultrasonic pings from the beacons. A beacon determines when to chirp by randomly selecting a delay between zero and one second. The beacon delays the required amount of time and then samples the RF channel for activity. If the beacon detects that the

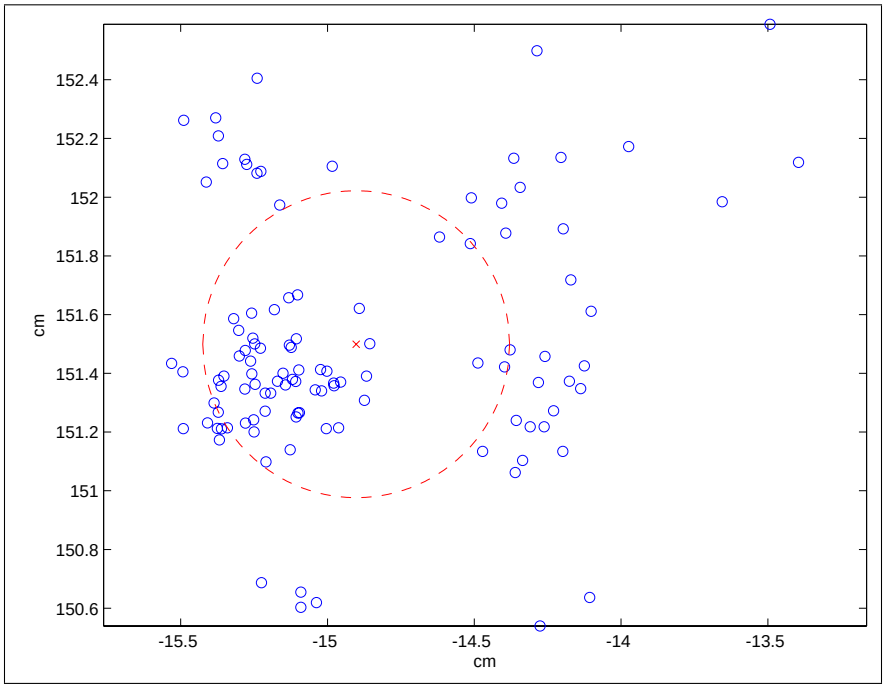


Figure 5.8: Static performance of the Cricket positioning system. The circle represents a CEP of .52 cm

channel is in use (by measuring RSSI), the beacon backs off a random amount of time before attempting a retransmit.

There are several problems with this system. First, there is the possibility for more than one ultrasonic chirp to be in transit at one time. Although an effort is made to disambiguate the source of a chirp, there is still the possibility for received ultrasound to be associated with the wrong beacon. There is also the possibility for two chirps to interfere with each other and cause range errors.

Another major problem is that the range measurements occur at different times. A listening unit that is moving will be unable to accurately determine its position from the range measurements alone. Even at modest velocities of 1 m/s, the individual range measurements might occur at listener locations that are spaced

0.3 m apart. The only way to achieve accurate position estimates while moving is to supplement the range measurements with information about the listener's motion. Of course, this greatly increases the system complexity.

The MIT implementation of Cricket may be used to compute position estimates in static and quasi-static environments. We should note that the listener unit does not run a position determination algorithm and does not account for cone angle errors. The unit simply outputs computed ranges that occur roughly once per second for each beacon in range. One advantage of the MIT system is that listener privacy is respected since no information is ever transmitted by the listener. The system is good for coarse positioning, when the user needs only to determine the closest beacon. For precise positioning on a moving vehicle, however, the MIT system falls short.

Chapter 6

Results

6.1 Simulated Results

6.1.1 Autopilot Simulink Model

Prior to testing the hovercraft autopilot on real hardware, we developed a high-fidelity model of the system using Simulink. We added blocks to model the INS noise, Bluetooth network delay, ESC reverse-to-forward delay, and actuator saturation. We also implemented the thruster and rudder calibration tables to convert between forces and actuator commands.

We implemented the hovercraft dynamics as a continuous-time S-function. In addition, we implemented the discrete-time hovercraft control laws as an independent *controller* block using standard Simulink components. By closing the loop around these two blocks, we created an *ideal* autopilot model for comparison.

The Simulink models greatly aided in debugging the controller block and determining appropriate gains for the real autopilot system. Once a desired level of simulated performance was attained, the controller block was tested on real hardware. The controller block was simply copied to the dSPACE autopilot model and compiled to C code.

Figure 6.1 shows the complete autopilot model used for simulation. Observe that the real and ideal models are connected in parallel allowing a direct comparison of performance. In the following sections, we give an overview of the functionality provided by each of the blocks.

INS Block

The INS block is shown in Figure 6.2. The block contains two noise generators that add random errors to the true velocity and heading variables output by the hovercraft dynamics block. We model two sources of error in the INS block.

First, we assume that there are small time-varying errors in the INS bias estimates for the accelerometers and gyro. Both bias errors are modeled as zero-mean Gaussian random variables with variances given by 2.5 cm/s^2 for the accelerometers and $1°/s$ for the gyro. The integrator computes the velocity and heading errors that result from these bias errors.

In addition, we corrupt the gyro measurements with zero-mean white Gaussian noise with a variance of $.5°/s$. This noise represents the IMU sensor sampling noise. Finally, the sample and hold block outputs the navigation data at the 6 ms IMU sampling rate.

Downlink/Uplink Blocks

The downlink and uplink blocks are simple delay chains that model the mean Bluetooth link delay determined empirically. Using the data presented in Table 5.1,

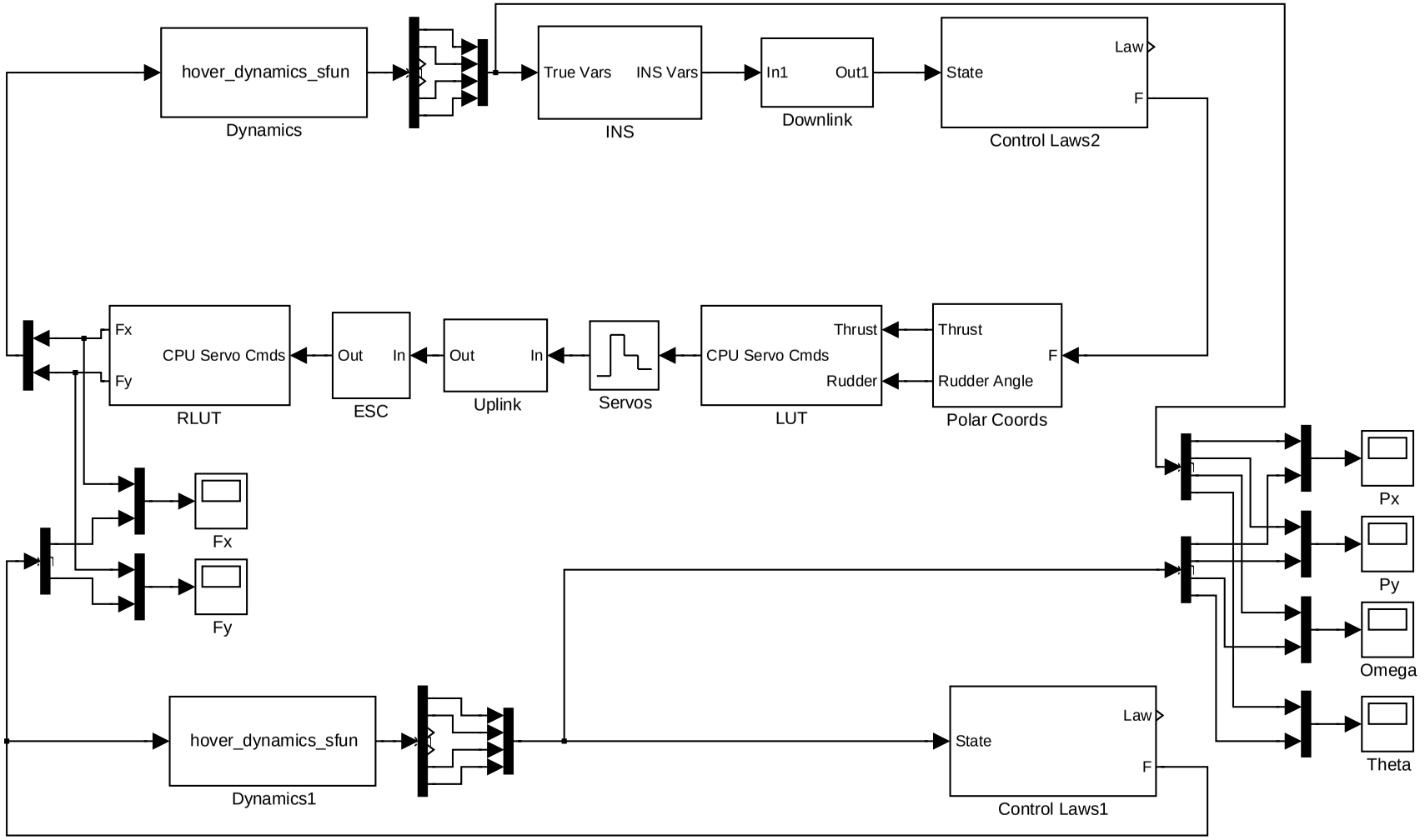


Figure 6.1: Simulink model of the hovercraft autopilot for simulation

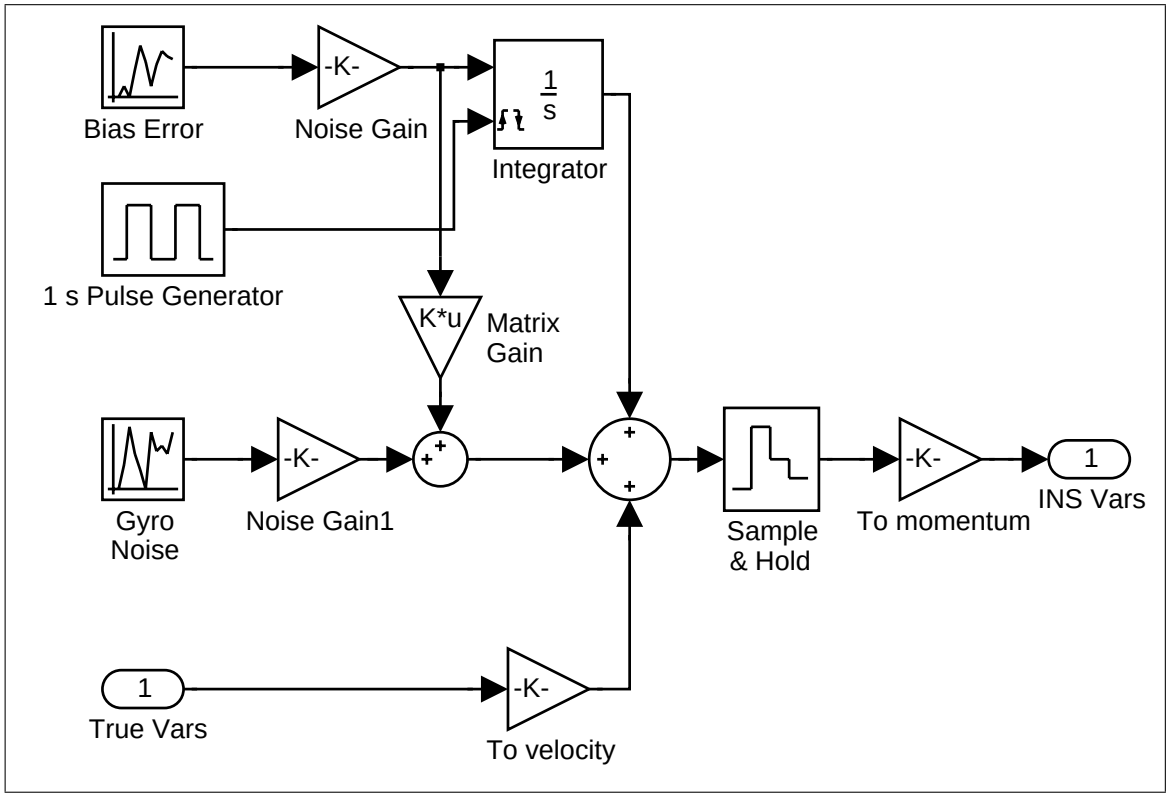


Figure 6.2: INS error model

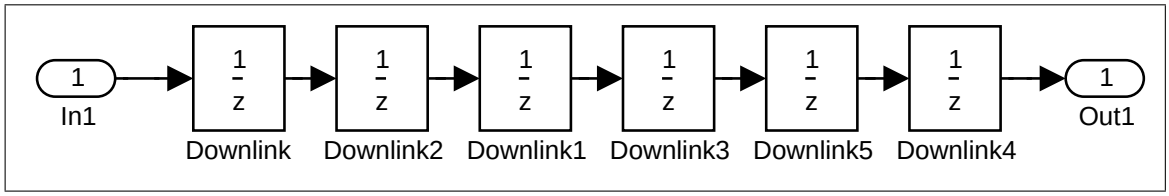


Figure 6.3: Hovercraft-to-controller (downlink) Bluetooth link delay

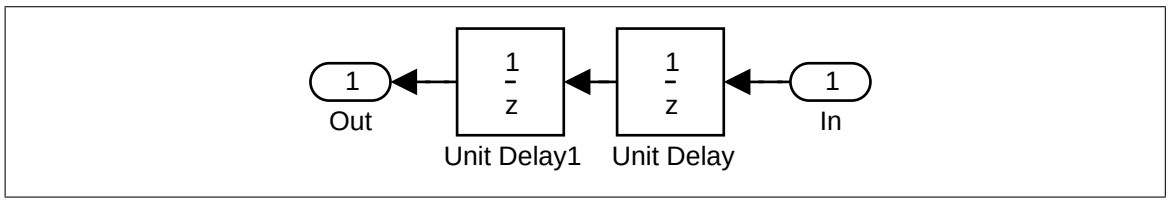


Figure 6.4: Controller-to-hovercraft (uplink) Bluetooth link delay

we see that a 2.9/22.0 kbps link has a mean delay of 40 ms up and 34 ms down. As shown in Figure 6.3, we model the downlink as a chain of 6 delay elements executing at the IMU rate of 6 ms. Similarly, Figure 6.4 depicts the uplink modeled by 2 delay elements. These delay elements run at the servo update rate of 20 ms.

LUT/RLUT Blocks

The LUT (lookup table) and RLUT (reverse lookup table) blocks convert forces to actuator commands and vice-versa. These blocks implement the rudder and thrust calibration tables discussed in section 4.3. Figure 6.5 shows the inside of the LUT block.

The LUT block works in tandem with the 'Polar Coords' block to output a rudder servo command in milliseconds and a thrust command in motor RPM.

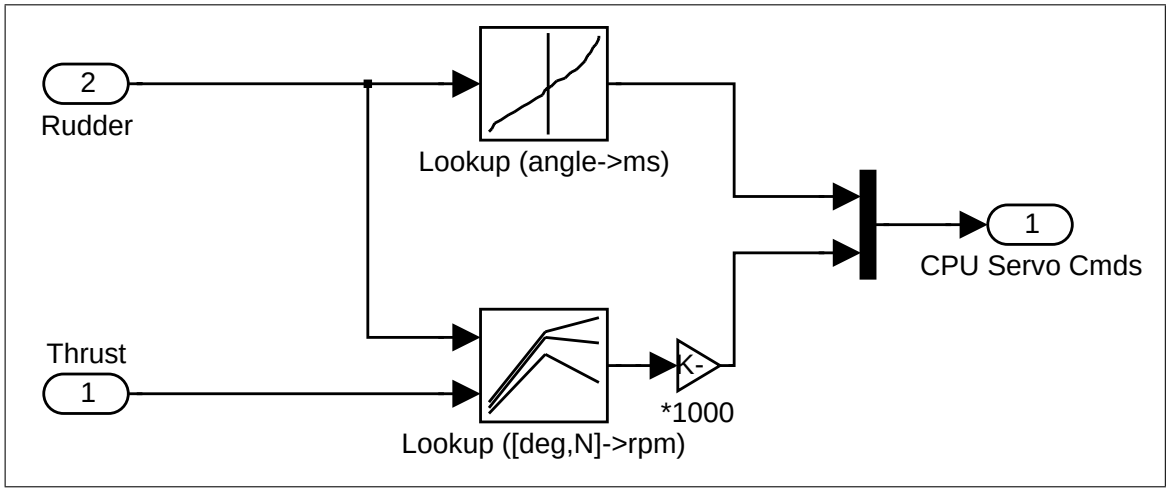


Figure 6.5: Actuator commands lookup table (LUT)

Actuator saturation is handled directly by the lookup tables. Commands that exceed the predefined actuator limits are clipped to their maximum values.

The 'Polar Coords' block precedes the LUT block and converts the F_X and F_Y force components to polar coordinates. It also models a force dead-zone nonlinearity of $[-.05, .05]$ N in order to prevent actuator chatter and conserve battery power. The rudder angle is then determined using equation (4.3).

The RLUT block takes a rudder servo command and a motor RPM as inputs and determines the forces that were actually imparted by the actuators. This block allows comparison of the realized forces with the forces requested by the controller and aids in tuning the control law gains.

Thrust Fan ESC Block

The ESC block models the reverse-to-forward thrust fan delay described in section 4.2.1. The block detects when the thrust fan RPM command crosses from

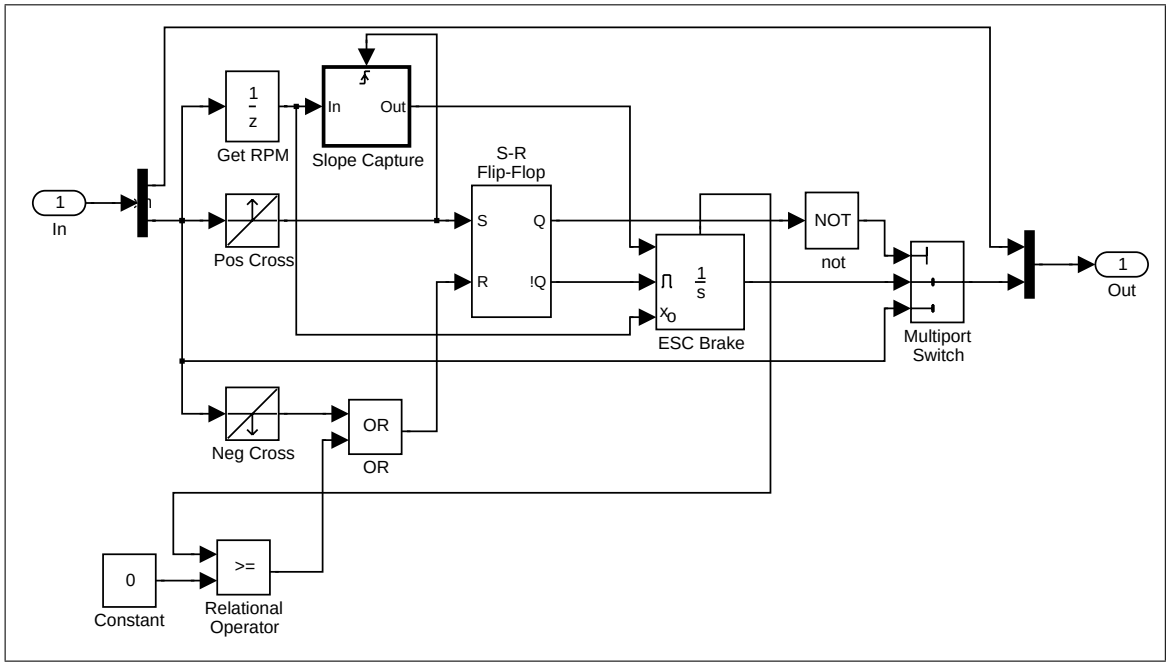


Figure 6.6: Electronic Speed Controller (ESC) model

negative to positive and then brakes the motor before switching directions. The braking interval lasts for exactly 150 ms. If a negative RPM command is received during the braking interval, the brake is reset and the fan jumps immediately to the commanded RPM. The ESC block is shown in Figure 6.6.

6.1.2 Zero Velocity Stabilization

Figures 6.7a - 6.7c shows the simulated results for zero velocity stabilization using the initial conditions and controller gains shown in Table 6.1.

The solid blue line represents the real autopilot and the dashed red line corresponds to the ideal autopilot. There are a couple of observations to make from these plots. First and foremost, the real autopilot model brings the hovercraft to rest! Despite all the delays and uncertainties present in the system, the controller

Parameter	Value
$V_X(0)$	.2 m/s
$V_Y(0)$	-.7 m/s
$\Omega(0)$	6.4 rad/s
k_1	1.0
k_2	2.5

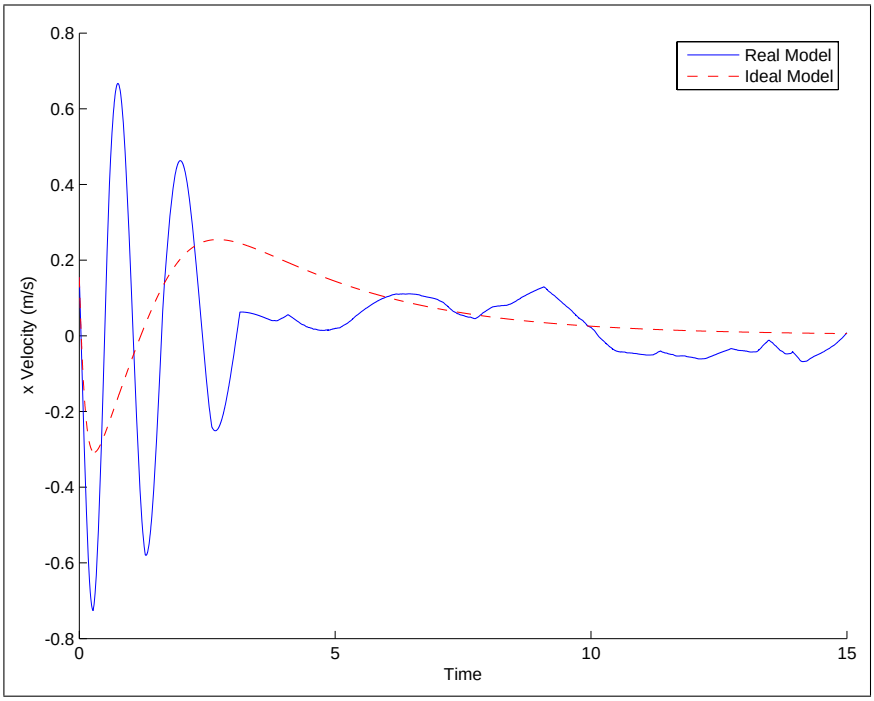
Table 6.1: Zero velocity stabilization parameters

effectively damps each of the velocities and requires only 5 seconds longer to stop the hovercraft turning. Second, the initial segments of the linear velocity plots for the real autopilot model are highly oscillatory with large overshoot compared to the ideal model. Finally, the settling time of the real model with the given initial conditions is roughly 10 seconds.

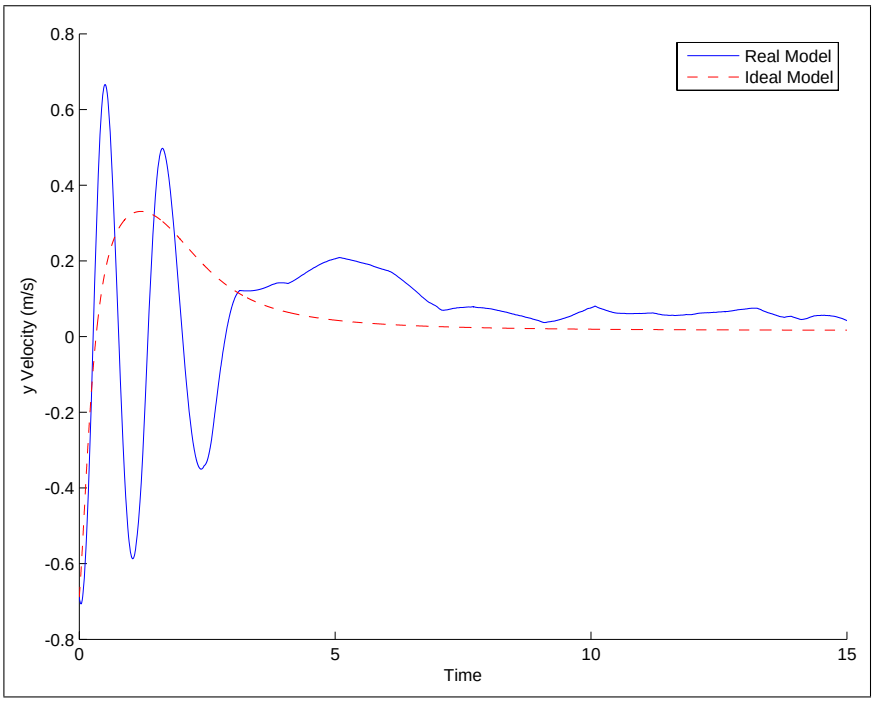
Considering that the real model accounts for a loop delay of nearly 80 ms, imposes realistic constraints on the actuator limits and bandwidth, and assumes error-prone INS outputs, the simulated performance of the real model is quite impressive. In section 6.2.2 we show that the simulated performance agrees well with the actual autopilot performance for the same initial conditions and controller gains.

6.1.3 Forward Velocity Stabilization

The next group of plots show the simulated autopilot performance for stabilizing a constant forward velocity. An arbitrary velocity of 1.1 m/s was chosen and the controller was started with the hovercraft at rest. The parameters selected for the simulation appear in Table 6.2.

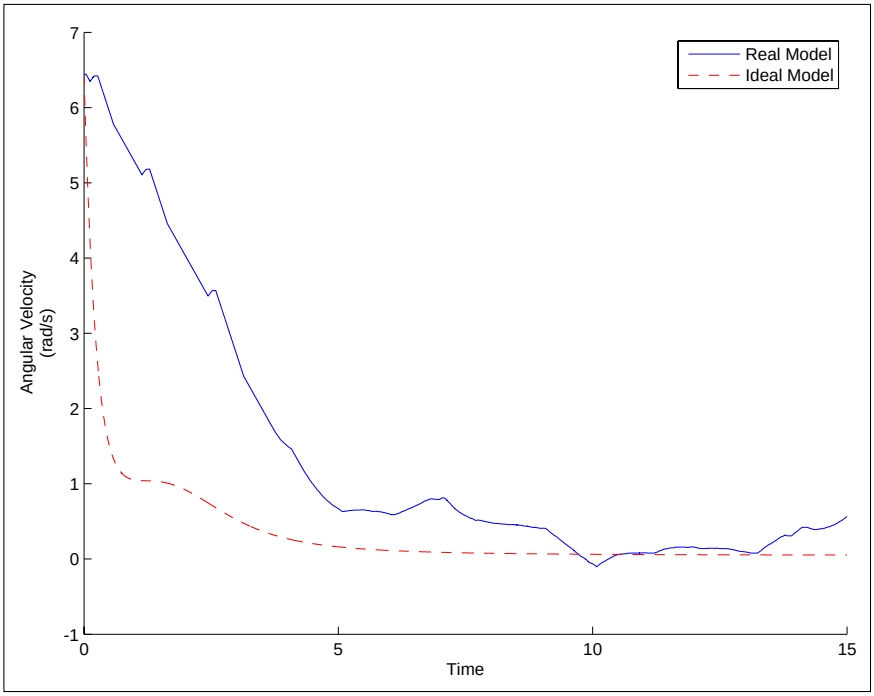


(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)

Figure 6.7: Simulated results for zero velocity stabilization



(c) Angular velocity (Ω)

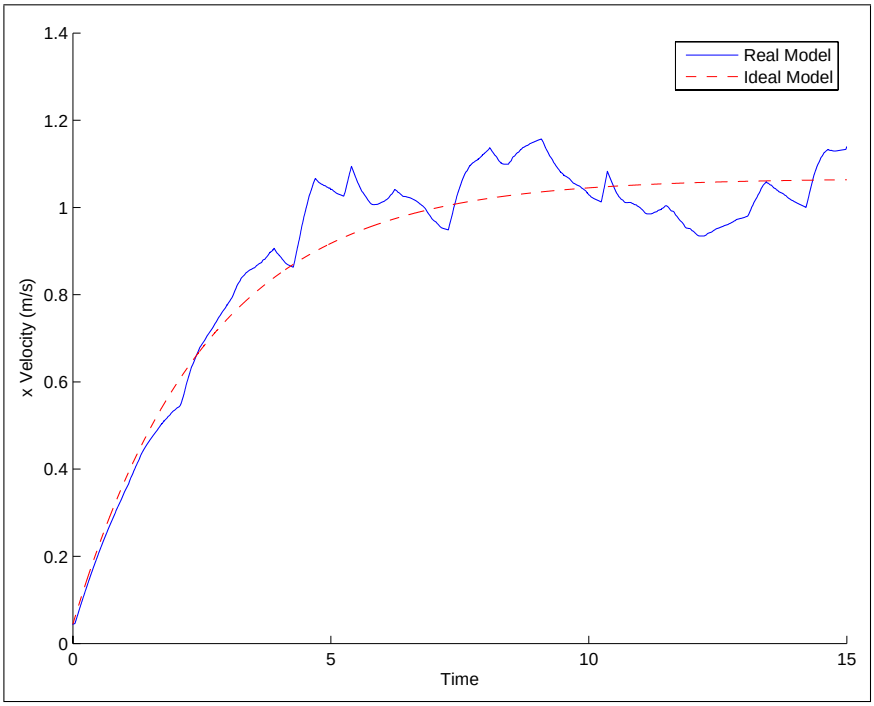
Figure 6.7: Simulated results for zero velocity stabilization

Parameter	Value
$V_X(0)$	0.0 m/s
$V_Y(0)$	0.0 m/s
$\Omega(0)$	0.0 rad/s
$\overline{P}_X$	2.8 kg m/s
k_1	1.0
k_2	4.0

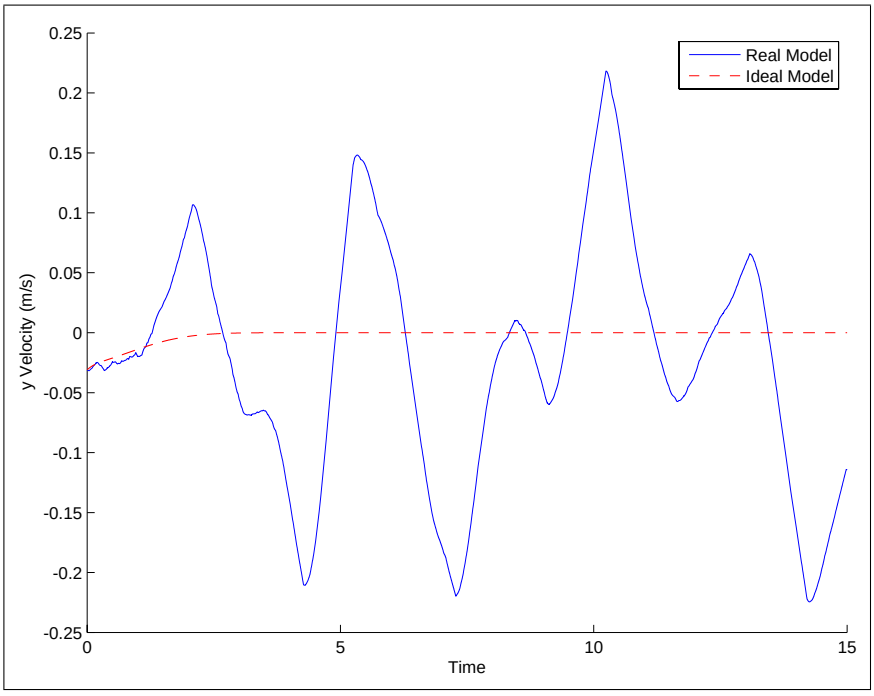
Table 6.2: Forward velocity stabilization parameters

Figures 6.8a - 6.8c depict the relevant state variables as the controller attempts to drive the hovercraft forward at a constant velocity of 1.1 m/s. The solid blue line represents the real autopilot performance and the dashed red line corresponds to the theoretical performance. It is clear from Figure 6.8a that both controllers achieve the commanded velocity. The real autopilot model attains the reference velocity for the first time in about 5 seconds, but the steady-state error fluctuates between $\pm.1$ m/s. The errors are due to the INS noise and delays present in the system. In contrast, the ideal model exhibits smooth asymptotic convergence to the desired reference velocity.

Figures 6.8b and 6.8c depict the lateral and angular velocities respectively. Observe that there is a marked difference between the real and theoretical autopilot performance exhibited in these plots. Although the real autopilot achieves average velocities of zero, the steady-state error fluctuates wildly as the controller contends with noisy state measurements and system delay. We have verified through simulation that the primary cause of these oscillations is the INS noise. Thus, the quality of the INS outputs is a deciding factor in the overall attainable performance.

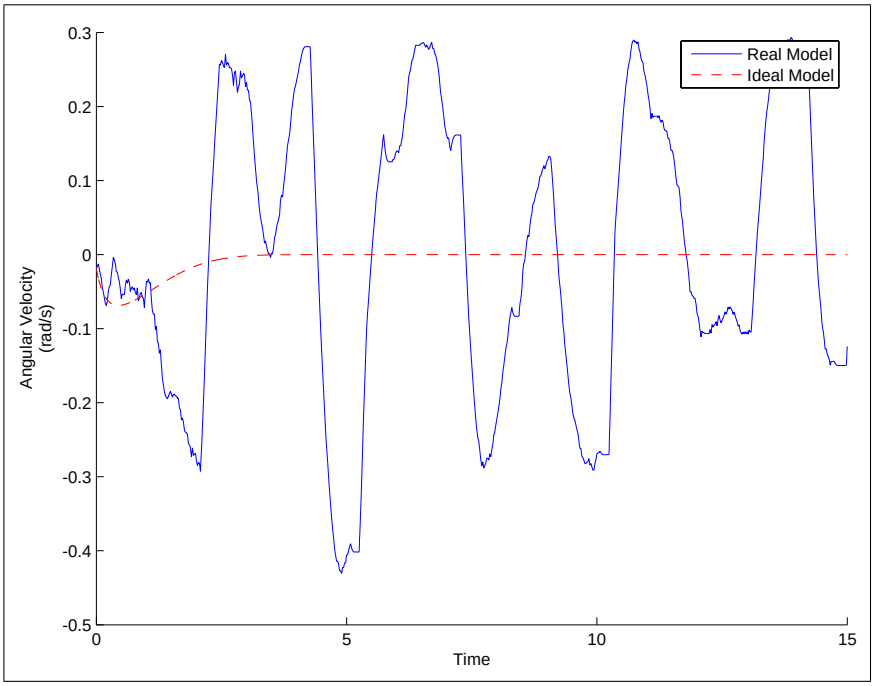


(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)

Figure 6.8: Simulated results for constant forward velocity stabilization ($\overline{V}_X = 1.1$ m/s)



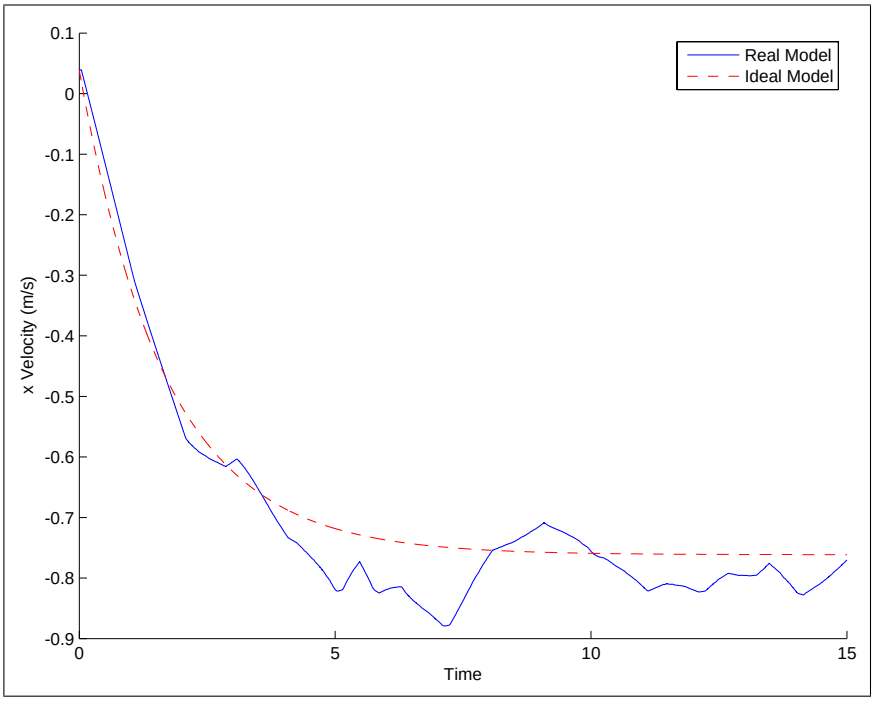
(c) Angular velocity (Ω)

Figure 6.8: Simulated results for constant forward velocity stabilization

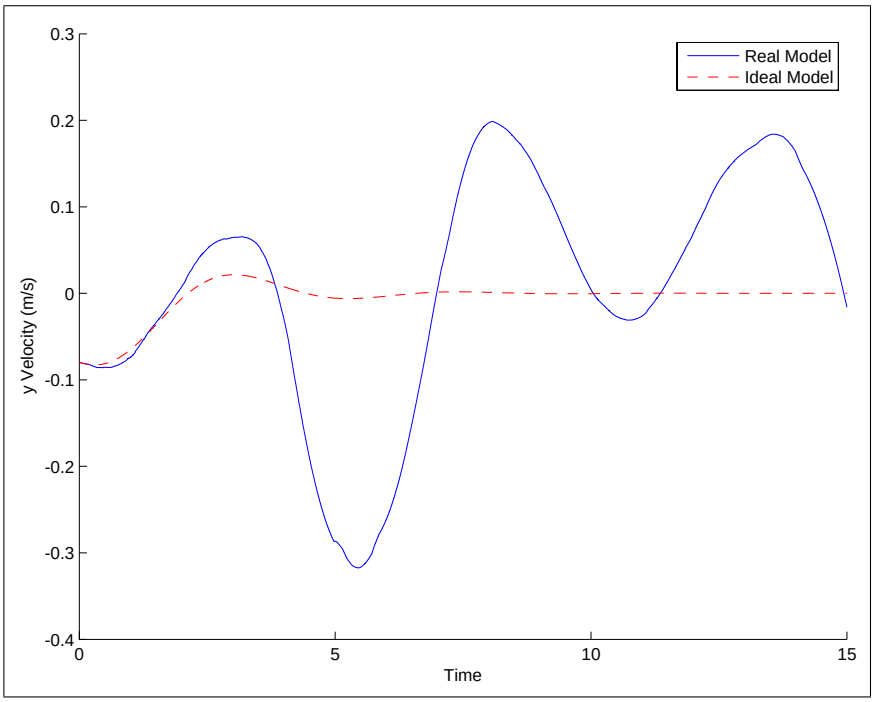
6.1.4 Reverse Velocity Stabilization

Figures 6.9a - 6.9c show the simulated results for reverse velocity stabilization. A reference velocity of -.76 m/s was selected and the autopilot was started with the hovercraft nearly at rest. The parameters used in the simulation appear in Table 6.3.

The plots strongly resemble the simulated forward velocity stabilization plots shown previously. In both cases, the lateral and angular velocities for the real system fluctuate about zero as the controller drives the longitudinal velocity to -.76 m/s. The controller achieves the desired reverse velocity in roughly five seconds.

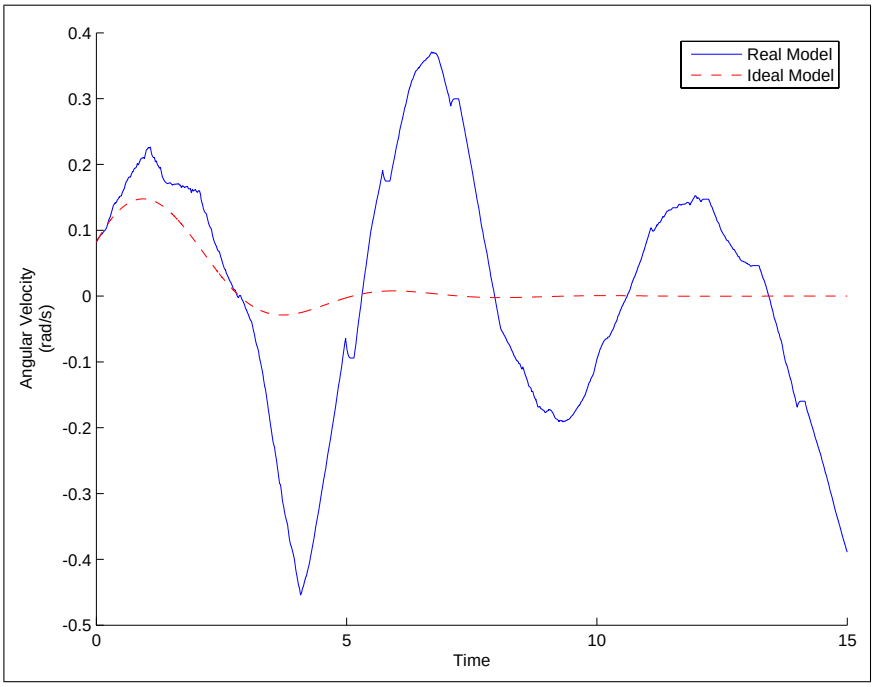


(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)

Figure 6.9: Simulated results for constant reverse velocity stabilization



(c) Angular velocity (Ω)

Figure 6.9: Simulated results for constant reverse velocity stabilization ($\overline{V}_X$ = -.76 m/s)

Parameter	Value
$V_X(0)$	0.0 m/s
$V_Y(0)$	-0.1 m/s
$\Omega(0)$	0.1 rad/s
$\overline{P}_X$	-1.98 kg m/s
k_1	1.5
k_2	1.0
β	1.44

Table 6.3: Negative velocity stabilization parameters

6.1.5 Constant Angular Velocity Stabilization

The simulated results for constant angular velocity stabilization are shown in Figures 6.10a - 6.10c. The controller was commanded to maintain an angular velocity of 3 rad/s. The parameters used in the simulation are presented in Table 6.4.

Parameter	Value
$V_X(0)$	0.0 m/s
$V_Y(0)$	-0.0 m/s
$\Omega(0)$	0.1 rad/s
$\overline{\Pi}$	.2880 kg m^2/s
k_1	0.5
k_2	3.5

Table 6.4: Angular velocity stabilization parameters

The plots show that the hovercraft must undergo some linear translation in order to commence turning. This is expected since the hovercraft dynamics preclude the production of a pure torque. Observe that the real and ideal autopilots achieve steady state angular velocity in roughly the same amount of time. The main difference is that while the state variables for the real autopilot oscillate about

the correct final values, the ideal autopilot exhibits asymptotic convergence. Thus, Figures 6.10a and 6.10b indicate that the real hovercraft will wander slowly while turning at the commanded velocity as a result of INS noise, delay, and other disturbances.

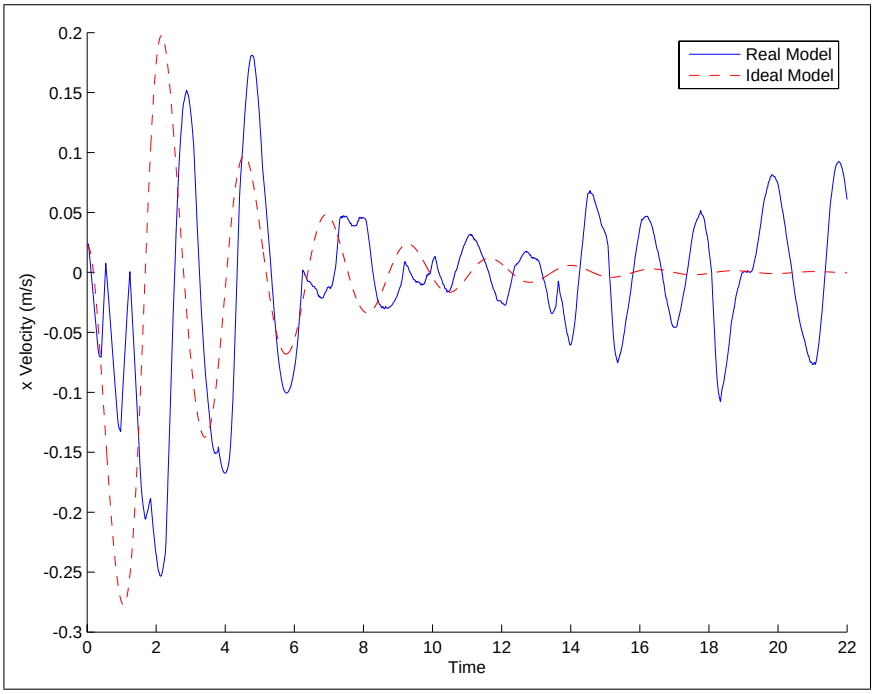
6.1.6 Heading Stabilization Comparison

We now present the simulated results for heading stabilization. We simulated both of the hybrid heading stabilization algorithms discussed in section 2.3.3 using two different sets of initial conditions. The results are plotted together to facilitate comparison of the different strategies.

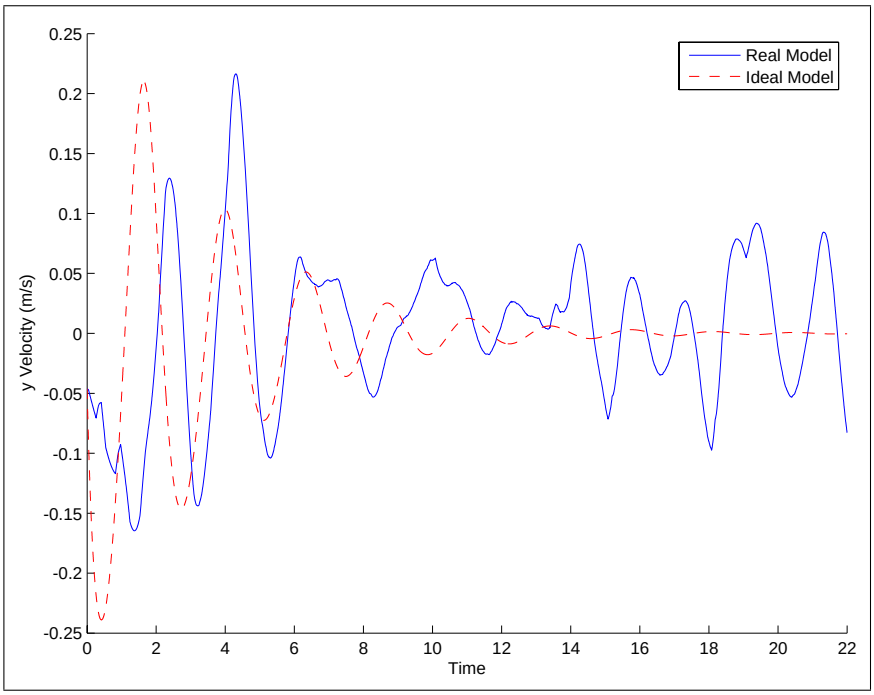
In the first trial, nonzero initial conditions were selected for the longitudinal, lateral, and angular velocities. The bang-bang pointing algorithm used a fixed angular velocity magnitude of 1.5 rad/s while the proportional law gain was set to 1.0. These values were selected to achieve a comparable transient response. Additionally, the desired heading, $\overline{\theta}$, was set to 5 rad and the pointing tolerance was 1°. Figures 6.11a - 6.11d show the simulated results.

The plots indicate comparable performance for the two pointing control laws. One noticeable difference is that the proportional law for the ideal autopilot exhibits steady state error. The cause of this error can be understood by examining the proportional heading stabilization algorithm.

When the hovercraft heading error satisfies the pointing tolerance, the autopilot switches to the zero velocity stabilization law. Due to the asymptotic nature of

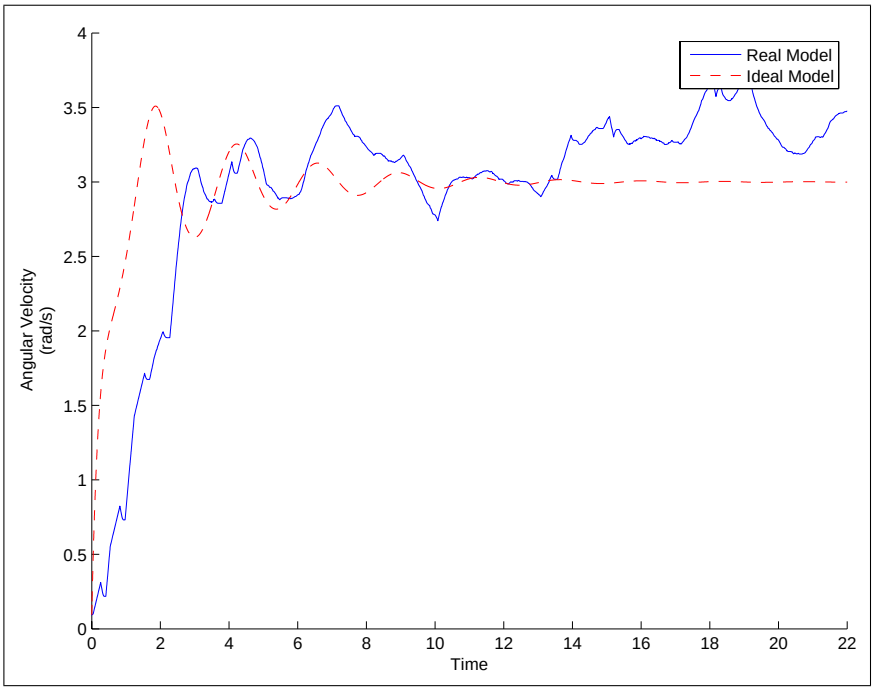


(a) Longitudinal velocity (V_X)



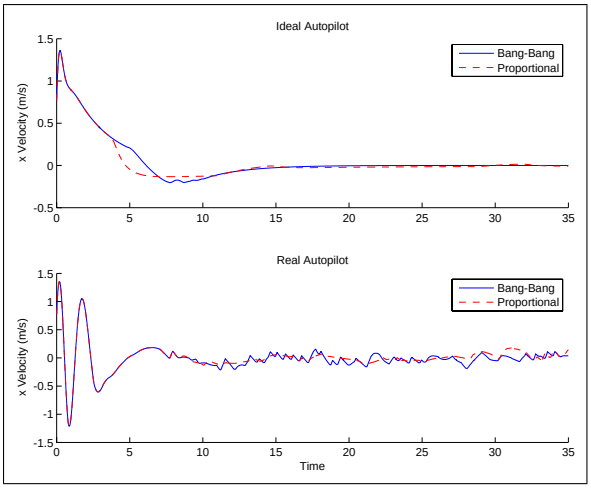
(b) Lateral velocity (V_Y)

Figure 6.10: Simulated results for constant angular velocity stabilization

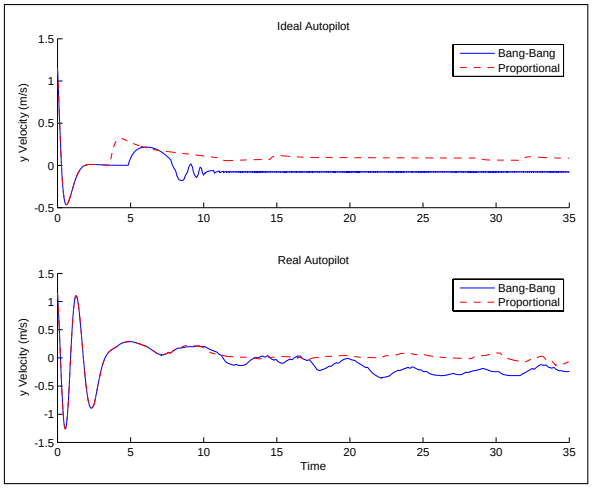


(c) Angular velocity (Ω)

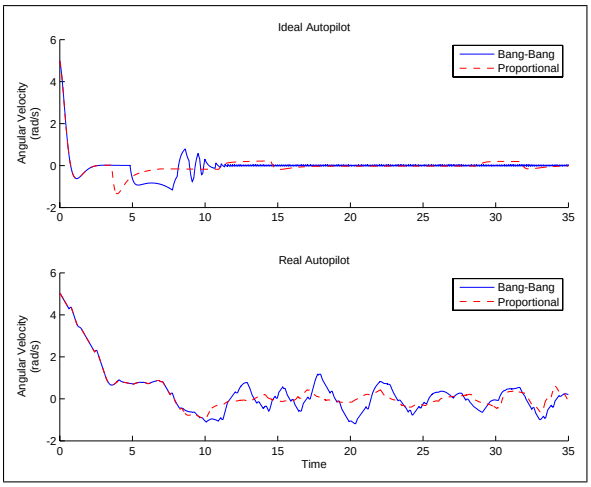
Figure 6.10: Simulated results for constant angular velocity stabilization ($\overline{\Omega} = 3$ rad/s)



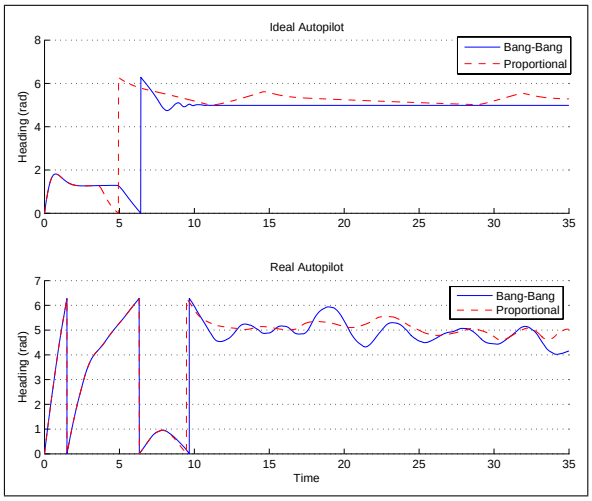
(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)



(c) Angular velocity (Ω)



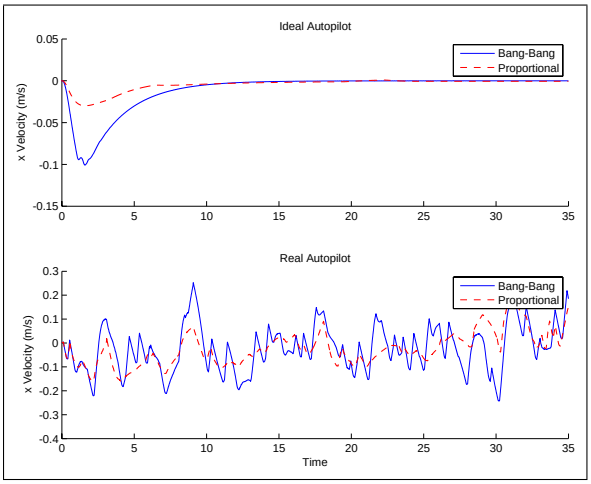
(d) Heading (θ)

Figure 6.11: Simulated results for heading stabilization comparison with nonzero initial conditions

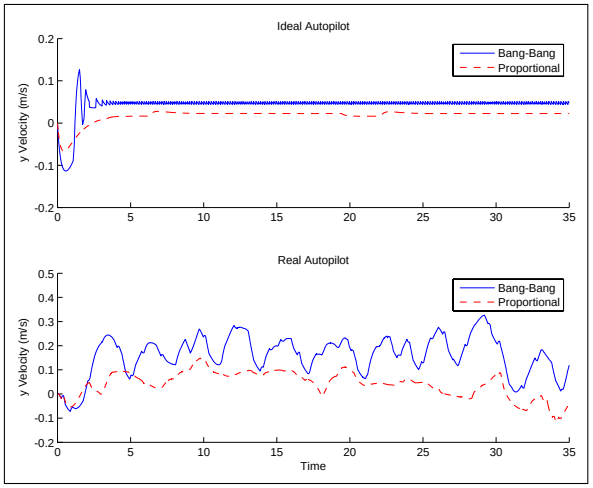
the velocity damping control law, the hovercraft continues to drift slowly. The problem is that when the heading error exceeds the pointing tolerance, the autopilot can not immediately switch back to the constant angular velocity stabilization law. The reason is that step (5) of the proportional heading stabilization algorithm prohibits switching control laws until condition (2.48) is satisfied. Since the desired angular momentum, $\overline{\Pi} = -k\,\hat{\theta}$, is small, condition (2.48) prevents the controller from switching to the $\overline{\Pi}$ stabilization law. Once the error signal becomes large enough, however, the controller is provisioned to switch laws and drive the heading error to zero. The bang-bang controller does not suffer from this problem because the selected $\overline{\Pi}$ is large enough to immediately satisfy (2.48) as soon as the heading tolerance is exceeded.

Figures 6.12a - 6.12d show the results for initial conditions equal to zero. The desired heading was set to 1 rad. All other parameters had the same values as in the previous simulation. Again, the plots indicate comparable performance between the two algorithms. In addition to achieving the desired heading more slowly, the proportional law exhibits steady state error as before. One advantage of the proportional law is that it drives the lateral velocity closer to zero. In contrast, the bang-bang algorithm chatters between the zero velocity and constant angular velocity control laws when the heading approaches the pointing tolerance. This chattering prevents V_Y from being driven to zero.

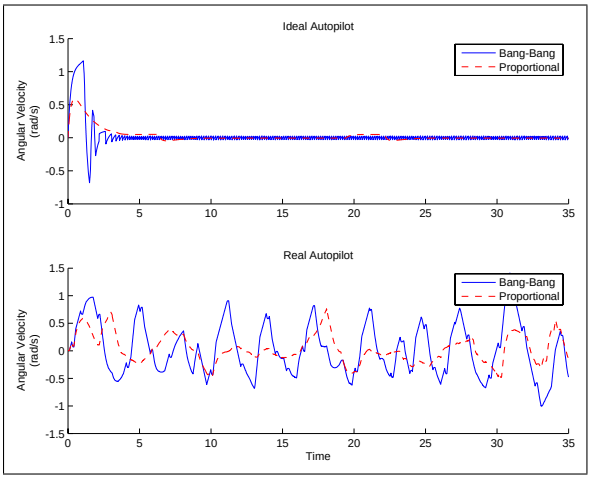
Based on these simulated results, it is still unclear whether to use the bang-bang or proportional heading stabilization algorithm on the real autopilot. On one hand, the proportional law seems to provide better stability at the expense of some



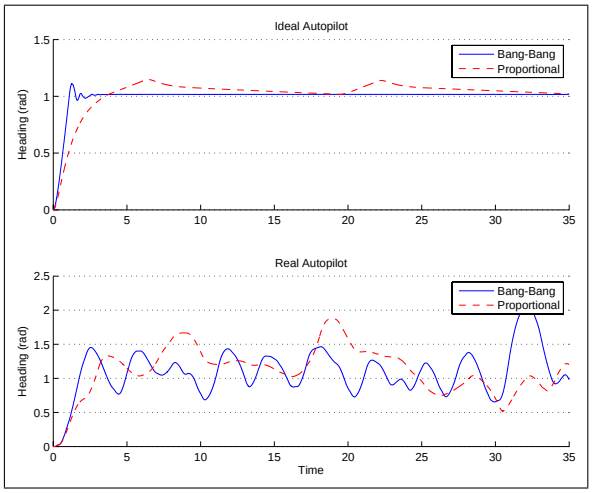
(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)



(c) Angular velocity (Ω)



(d) Heading (θ)

Figure 6.12: Simulated results for heading stabilization comparison with initial conditions equal to zero

steady state error. On the other hand, the questionable small performance benefit of the proportional law does not seem to justify the additional complexity of the algorithm.

6.2 Experimental Results

6.2.1 Aided INS Performance

6.2.1.1 Test Configuration

The aided INS performance was quantified using a rotating platform under computer control. The platform was driven by a high power DC supply with remote operation capability. Reference signals were provided by a 4096-position optical encoder connected to the motor shaft and an analog tachometer.

Prior to testing, the tachometer was calibrated by applying a slowly increasing ramp voltage to the motor and recording the tachometer output. The true angular velocity was computed by differentiating the position encoder data and applying heavy low pass filtering (4th order Bessel filter at 5 Hz). A linear regression was performed on the collected data to provide the mathematical relationship between the tachometer output voltage and the platform angular velocity.

The INS was tested in two separate configurations. For the first test, the IMU was mounted inline with the motor spin axis. This configuration was used to evaluate the heading filter performance. For the second test, the IMU was mounted on an aluminum beam attached to the platform and offset a distance of 12 inches

from the spin axis. This setup allowed the IMU to sense centripetal and tangential accelerations and was useful for quantifying the velocity/position filter performance.

Power and data lines to the IMU were routed through an 8-channel slip ring coupled to the motor shaft. A Simulink model to control the rotating platform was developed and executed in real-time on the dSPACE system. Data was collected by dSPACE, plotted in real-time using the ControlDesk data visualization application, and streamed to disk for post-processing in Matlab.

6.2.1.2 Heading Filter Performance

The heading filter was developed and tested first. To test the filter, the IMU was mounted inline with the motor spin axis to reduce any platform wobble that might adversely affect the results. For the first sets of tests, heading measurements were provided by the IMU magnetometer at an update rate of 6.5536 ms. Later, we repeated the tests using heading estimates provided by two Cricket units.

We first drove the platform at a constant angular velocity of -2.9 rad/s (-166°/s). Figures 6.13a and 6.13b compare the INS angular velocity and heading outputs with the reference signals. The INS outputs are shown in dashed red and the reference signals are in solid green. As seen in Figure 6.13a, the INS angular velocity tracks the tachometer output well and has a lower variance. There is a slight oscillation in both outputs which we attribute to small variations in the motor speed. The angular velocity error plot indicates that the error magnitude is small. In fact, the angular velocity rms error is .065 rad/s (3.7°/s). Also, as shown in Figure 6.13b the

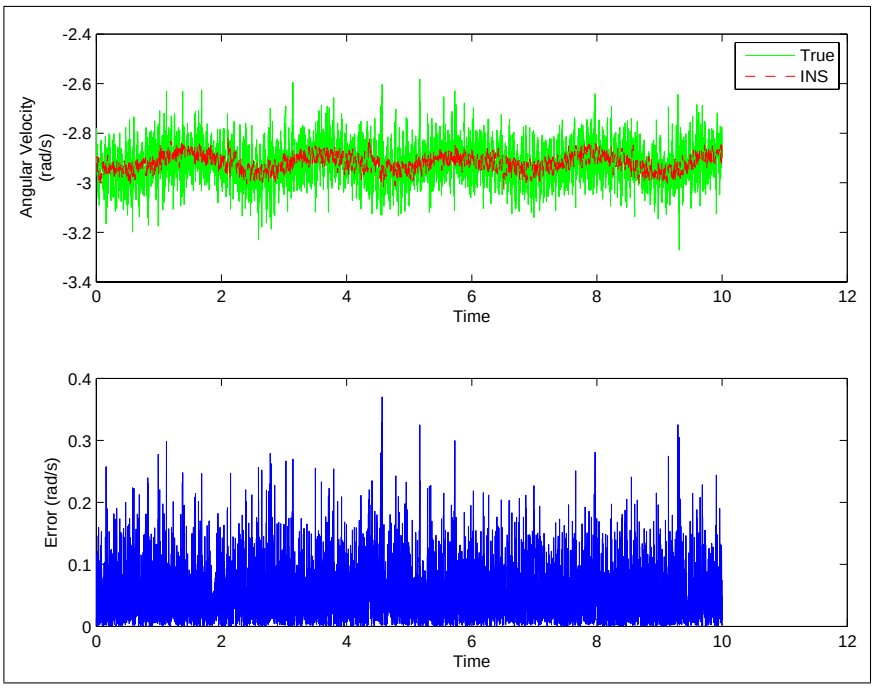
INS heading estimate agrees well with the optical encoder output. The heading rms error is .009 rad (.5°).

In a subsequent test, we increased the velocity of the platform and evaluated the filter performance. Even as the platform angular velocity approached the gyroscope limits of ±300°/s, the filter performed well. In fact, when the angular velocity was increased beyond the device limits, the filter adjusted the bias estimate to compensate.

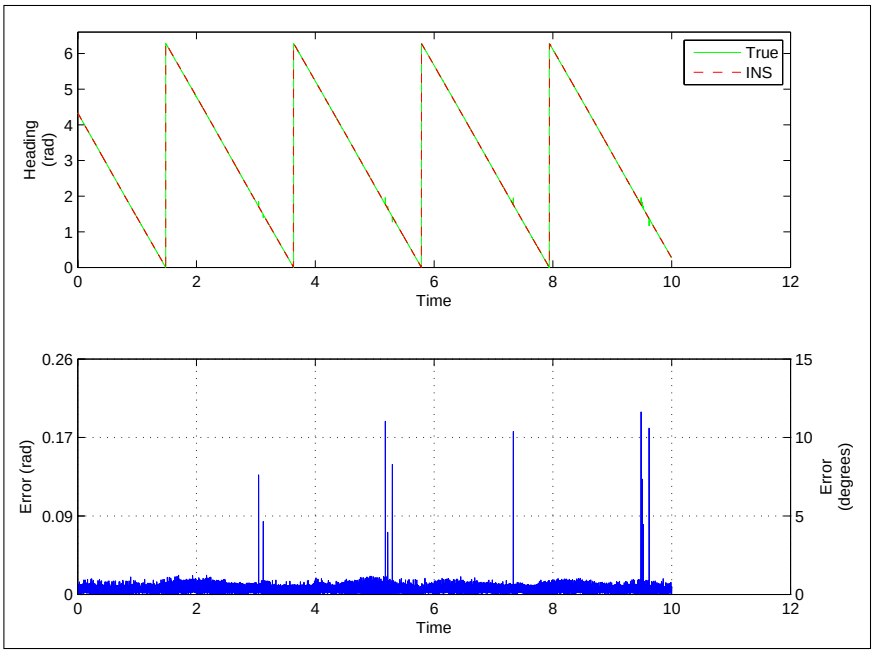
We repeated this constant angular velocity test using two Cricket units as the heading aiding source. The two units were separated by 8 inches. One Cricket device was mounted adjacent to the IMU and one was secured to an aluminum beam connected to the motor shaft. Heading estimates were provided once per second.

We tested the INS at three different angular velocities (-57, -115, -230 °/s). Figure 6.14a compares the INS angular velocity estimate with the tachometer reference. The INS output is the dashed red line and the tachometer is plotted in solid green. It is clear that the INS tightly tracks the tachometer reference signal with minimal error. The error variance becomes slightly larger as the platform angular velocity is increased. For example, with the platform spinning at -115 °/s, the angular velocity rms error is .07 rad/s (4.1°/s). Observe that the rms error is similar to the error obtained using the magnetometer.

Figure 6.14b shows the INS heading estimate in dashed red compared with the optical encoder output in solid green. The discontinuities in the error plot indicate where the Cricket heading measurements are incorporated by the filter. Observe that the heading error increases from about 5 to 20 degrees as the platform spins



(a) Angular velocity (Ω), rms error = .065 rad/s (3.7°/s)



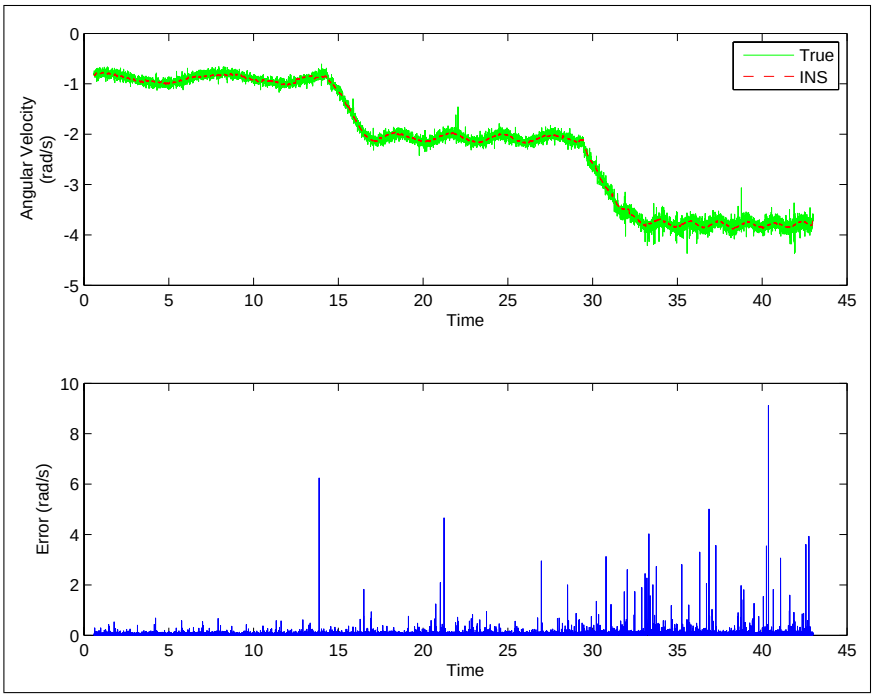
(b) Heading (θ), rms error = .009 rad (.5°)

Figure 6.13: Magnetometer-aided INS heading filter performance

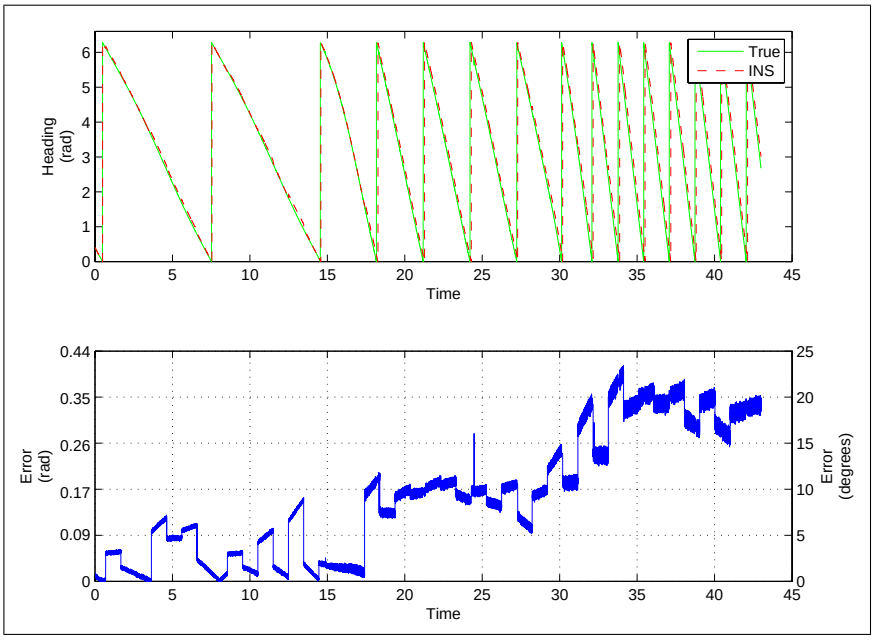
faster. For a platform angular velocity of -115 °/s, the heading rms error is .17 rad (9.8°). The angular velocity and heading errors are largely due to Cricket latencies as explained below.

Recall from section 5.5 that each Cricket unit is allocated a 100 ms time slice in which to obtain range information from the network. Within the 100 ms window, there is a small delay from when the unit pings the network to when it receives range measurements from the beacons. There is also a small data processing and transmission delay. As a result, the position error is dependent on the ground speed of the unit. An additional complication arises when two Cricket units are used to obtain heading estimates, as explained next.

When measurements from two Cricket units are available, the angle of the line joining the two devices may be computed. If the units are aligned along the body frame X axis, then the orientation of this line corresponds to the body heading in the inertial frame. There are two primary sources of error when using real Cricket devices for heading estimation. First, the position estimates are noisy. One way to reduce the effect of sensor noise on heading error is to separate the units as much as possible. Second, there is a fixed 100 ms delay between position estimates from the units. If the Cricket devices are moving, then the first unit's position estimate will be inaccurate when the second unit computes its position 100 ms later. As a result, the heading error becomes larger as the angular velocity increases. Later, we will show how the INS position outputs may be used to increase the accuracy of the heading measurements.



(a) Angular velocity (Ω), rms error = .07 rad/s (4.1°/s)



(b) Heading (θ), rms error = .17 rad (9.8°)

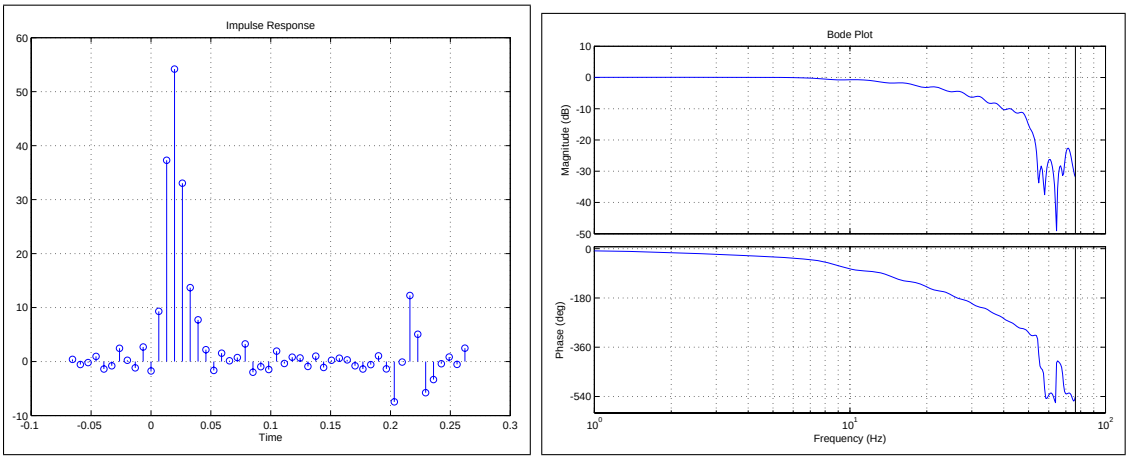
Figure 6.14: Cricket-aided INS heading filter performance. (rms values given for a platform angular velocity of -115 °/s.)

The second set of tests were designed to evaluate the heading filter under dynamic conditions. These tests were conducted with the magnetometer aiding source only. We began by applying a 5 Hz sinusoidal oscillation to the platform. There was a small phase lag in the INS angular velocity which became more pronounced as the frequency was increased. The phase lag was accompanied by some amplitude attenuation. At 20 Hz, the phase shift was nearly $-45°$, and the amplitude was attenuated by roughly 10% (-.92 dB). The INS heading also exhibited phase lag, although to a lesser extent.

Accurate attitude (heading) estimates are crucial to achieving good INS performance. Recall that vehicle orientation determines the rotation matrix used to resolve body vectors in the inertial frame. For a two-dimensional INS, a small constant error in heading produces an error term that grows linearly in the velocity estimate and quadratically in the position estimate. Furthermore, accurate heading estimates are needed in order to decouple the INS error equations in (3.27) and use the dual Kalman Filter approach discussed in 3.3.3.3. In the next section, we determine an appropriate dynamical model for the gyroscope with the end goal of increasing the accuracy of the Kalman Filter heading estimate.

6.2.1.3 Gyro Frequency Response and Compensation

We determined the frequency response of the gyroscope by performing a frequency sweep on the rotating platform. Sinusoids of 5, 10, 15, 20, 30, 40, and 50 Hz were applied for 3 seconds each. We used the Matlab System Identification Tool-



(a) Gyro impulse response

(b) Gyro frequency response

Figure 6.15: Gyro dynamic response

box to analyze the collected data and perform parametric model estimation. In the analysis, the tachometer signal was treated as the input to a linear time-invariant system, and the gyroscope data constituted the output.

The System Identification Toolbox provides several algorithms for determining system model coefficients from empirical data. We first performed a preliminary spectral and correlation analysis on the collected data. This analysis yielded the frequency and impulse response estimates shown in Figures 6.15a and 6.15b below.

Observe that the gyroscope frequency response exhibits significant dynamical effects. Furthermore, the gyro impulse response plot suggests that there are pure sample delays present in the system.

We used the following output-error form to model the dynamics:

$$y(n) = \frac{B(z)}{F(z)}u(n) + e(n) \tag{6.1}$$

where $B(z)$ and $F(z)$ are polynomials in z, $H(z) = \frac{B(z)}{F(z)}$ is a proper transfer function, and $e(n)$ is the disturbance process. The fitting algorithm selected a model with a 2 sample delay and a transfer function with three stable poles and one non-minimum phase zero. While the model fit the experimental data well, the inverse transfer function could not be physically realized. The inverse transfer function was non-causal and unstable due to the sample delay and non-minimum phase zero.

Unfortunately, the System Identification Toolbox does not support a minimum phase constraint on models. In order to obtain a stable invertible transfer function, we first removed the pure sample delay by time-advancing the gyro data. We then restricted the search space of the fitting algorithm. Specifically, we constrained the model to have no delay and zeros only at the origin. The Toolbox returned the following second-order model:

$$y(n) = \frac{.3236z^2}{z^2 - .9805z + .2898}u(n) \tag{6.2}$$

Figure 6.16 compares the frequency response of the LTI model specified in (6.2) with the gyroscope frequency response determined experimentally. Observe that both the magnitude and phase agreement is quite good for frequencies below 20 Hz. Between 20 and 50 Hz, the model does not exhibit enough phase lag and has too much amplitude attenuation. These errors will cause overcompensation in gain and insufficient phase lead in the high frequency components applied to the inverse filter.

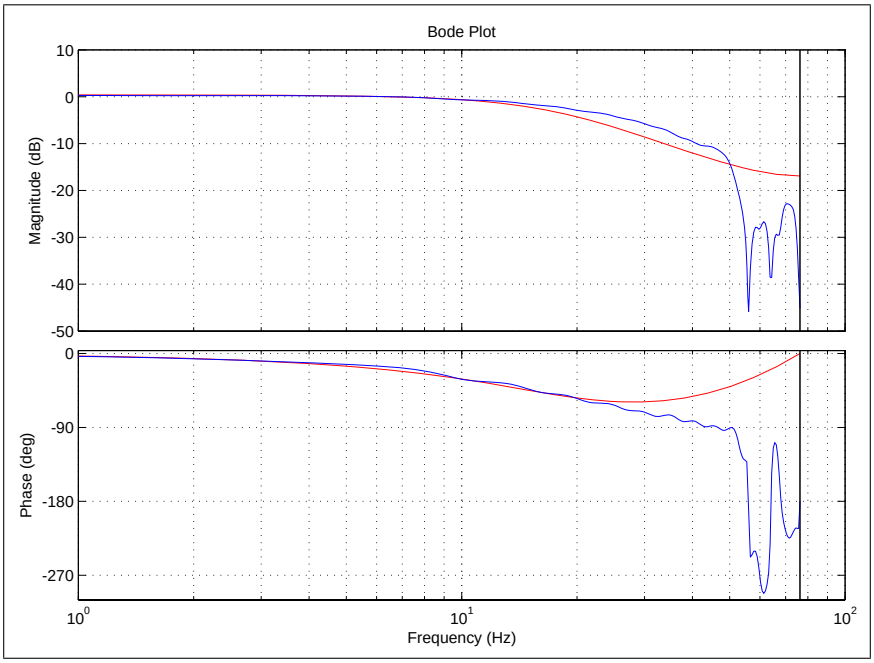


Figure 6.16: Experimental gyro frequency response (blue) and gyro model frequency response (red)

We implemented the inverse of transfer function (6.2) as a discrete block in Simulink and collected data to compare the compensated and uncompensated performance. Table 6.5 compares the results obtained by applying various sinusoidal inputs to the platform and measuring compensated and uncompensated gyro performance. As before, the tachometer output was taken to be the true angular velocity. The gain and phase shift for each data set was determined experimentally by using FFTs to compute the frequency response.

Ideally, each row in the compensated column should have unit gain and zero phase shift. The fifth column, Δt, indicates the reduction in time delay from applying compensation. Thus, improvement is on the order of one sample period (6.5536 ms) for frequencies up to 20 Hz.

	Uncompensated		**Compensated**		
Freq (Hz)	Gain	Phase (rad)	Gain	Phase (rad)	Δt(ms)
5	1.05	-.65	1.03	-.38	-8.59
10	0.98	-1.05	1.06	-.51	-8.59
15	0.93	-1.95	1.20	-1.17	-8.28
20	0.90	-2.39	1.48	-1.45	-7.48
30	0.67	-3.55	1.80	-2.52	-5.46
40	0.47	-4.61	1.87	-3.70	-3.62
50	0.33	-5.29	1.74	-4.58	-2.26

Table 6.5: Gyro performance with and without frequency compensation

The price paid for smaller time delay is an amplification of the high frequency noise components present in the sensor readings. The improved phase response, however, outweighs the small additional noise incurred. Since the INS will ultimately be used in an autopilot application, a reduction in phase lag will increase the closed-loop stability.

We designed a test to evaluate the aided INS performance with and without gyro frequency compensation. We first created a Simulink model with two instances of the heading filter. One filter received frequency compensated gyro data and the other was connected directly to the raw gyroscope output. Heading corrections were provided by the magnetometer. We then conducted a worst-case test that involved spinning the IMU at a high rate for three seconds followed by an abrupt stop. The platform was then spun in the reverse direction for an additional three seconds. This square wave input was applied for 25 seconds, and data was collected from both heading filters.

Figure 6.17 shows a close-up of the angular velocity waveform produced during the test. The true angular velocity is plotted in solid blue, the compensated angular

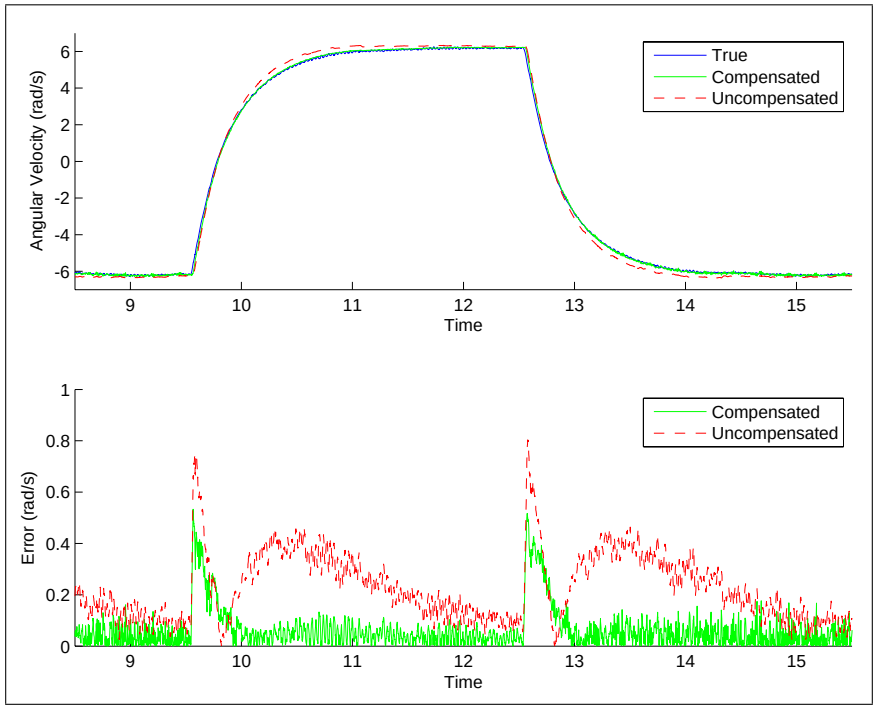


Figure 6.17: Comparison of INS angular velocity estimate with and without gyro frequency compensation

velocity is shown in solid green, and the uncompensated output appears in dashed red. The benefit of gyro frequency compensation is clearly visible from the error plot. The uncompensated gyro lags both the tachometer and frequency-compensated gyro waveforms during the transients. The lag appears as a spike on the error plot which gradually decays as the platform achieves a constant angular velocity.

Finally, Table 6.6 compares the root mean square error of the compensated and uncompensated angular velocity and heading measurements using the square wave input. The data indicates a small reduction in both angular velocity and heading error by using gyro frequency compensation.

	Compensated	Uncompensated
Angular Velocity	0.20 rad/s (11.5°/s)	0.30 rad/s (17.2°/s)
Heading	0.019 rad (1.1°)	0.044 rad (2.5°)

Table 6.6: Heading filter rms error comparison

6.2.1.4 Velocity/Position Filter Performance

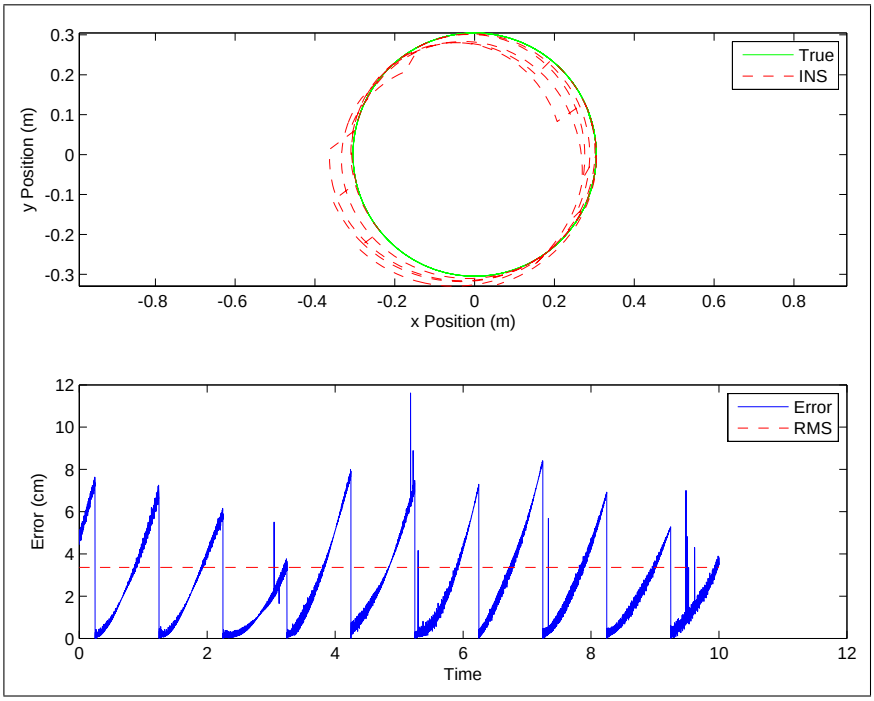
The INS velocity and position filter performance was also evaluated using the rotating platform. For this test, the IMU was mounted on an aluminum beam and offset 12 inches from the motor shaft. The IMU X axis was aligned perpendicular to the radial direction and the Y axis pointed inward along the radial direction. This configuration allowed the IMU X and Y-axis accelerometers to measure the tangential and centripetal accelerations respectively. As before, the test involved spinning the platform at a constant angular velocity and comparing the true and INS-computed quantities.

The first test used the magnetometer and optical encoder as the heading and position aiding sources. Heading estimates were provided at the IMU sample rate of 6.5536 ms and position updates were provided once per second. In the second test, the INS was aided exclusively by Cricket devices. Heading was computed using data from two Cricket units while position estimates were obtained from a single device. The heading and position measurements were provided once per second. Both tests used the optical encoder as the true heading and (x, y) position reference. In addition, the tachometer was used to compute the true linear velocity and acceleration vectors in the body frame. These vectors were then resolved in the inertial frame using the true heading provided by the optical encoder.

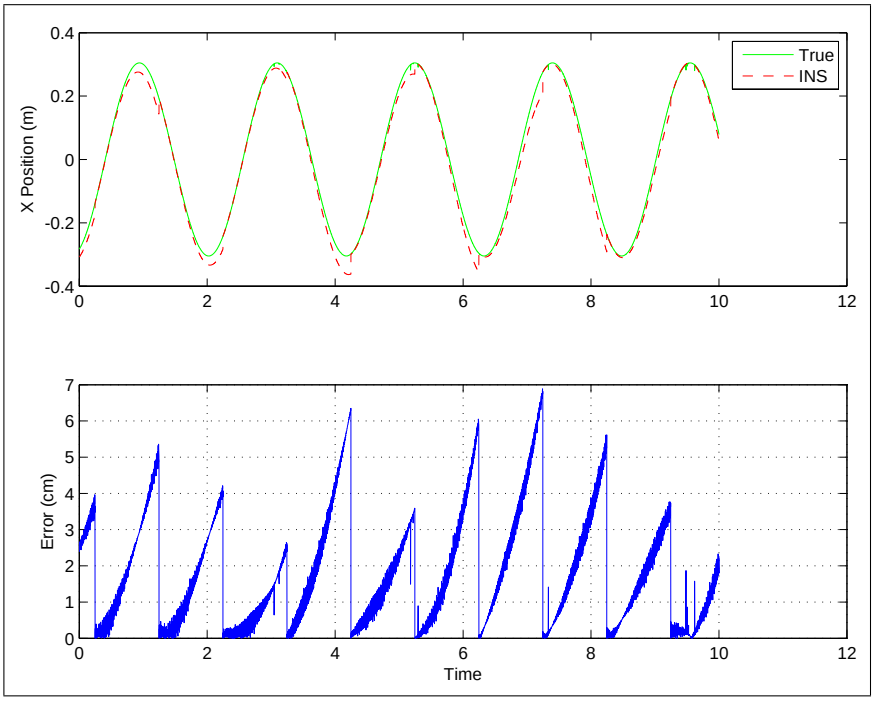
Prior to testing, we computed a fixed offset (in optical encoder counts) to align the magnetometer heading to the optical encoder output. Figures 6.18a - 6.18c show the test results for the INS aided by the magnetometer and position encoder. The true quantities are shown in green and the INS outputs are plotted in dashed red. During the test, the platform was spun at a constant angular velocity of -166°/s.

Figure 6.18a clearly shows the circular path traced by the IMU. Observe that the error grows quadratically between position updates due to small errors in the estimated accelerometer biases. When a new position update is available each second, the error resets to zero. Figures 6.18b and 6.18c show the x components of position and velocity in the inertial frame. The impulsive position corrections are easily discerned from these plots. We computed the position rms error to be 3.4 cm and the velocity vector rms error to be 7.5 cm/s.

Next, we repeated the constant angular velocity experiment using the Cricket devices as the aiding sources. Heading and position measurements, if available, were provided each second. Figure 6.19a depicts the IMU position in the inertial frame for a platform angular velocity of -115 °/s. The black plus signs mark the Cricket position measurements. The position rms error is 8.1 cm and the velocity vector rms error is 16.8 cm/s. In contrast to Figure 6.18a, the position error does not reset to zero each time a new position measurement is available. In addition, Figures 6.19b and 6.19c show that the position and velocity errors become larger as the angular velocity is increased. These errors result from fixed Cricket latencies, as described in section 6.2.1.2.

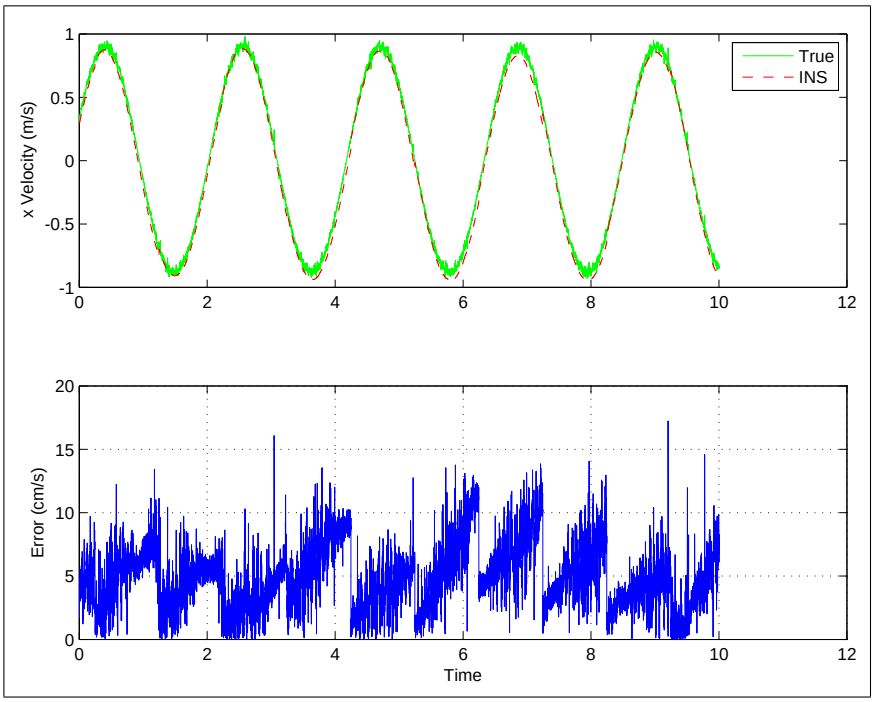


(a) Position (cm), rms error = 3.4 cm



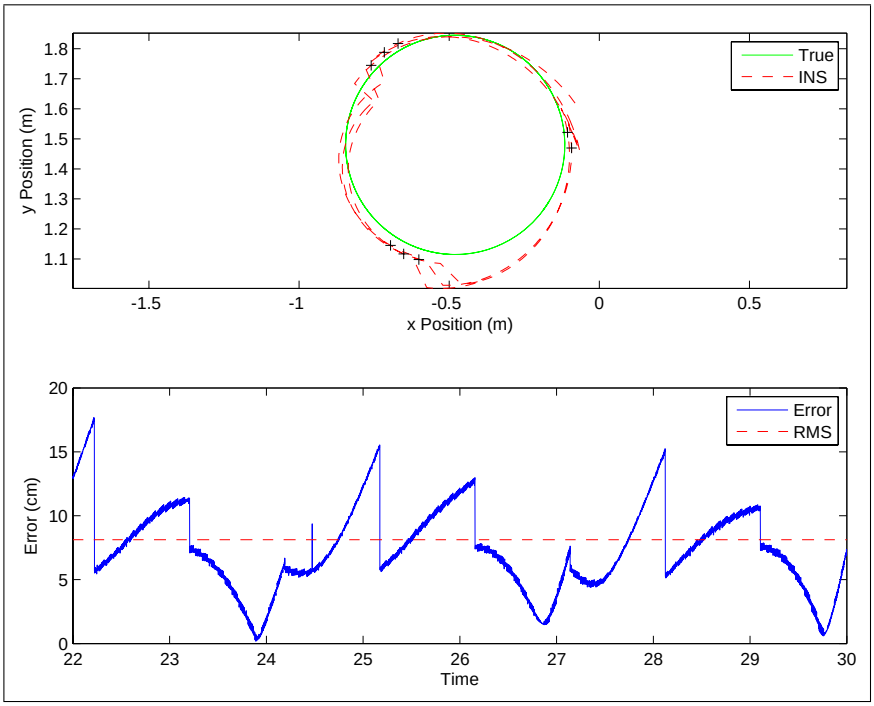
(b) x Position (m)

Figure 6.18: Magnetometer and optical encoder-aided INS velocity/position filter performance

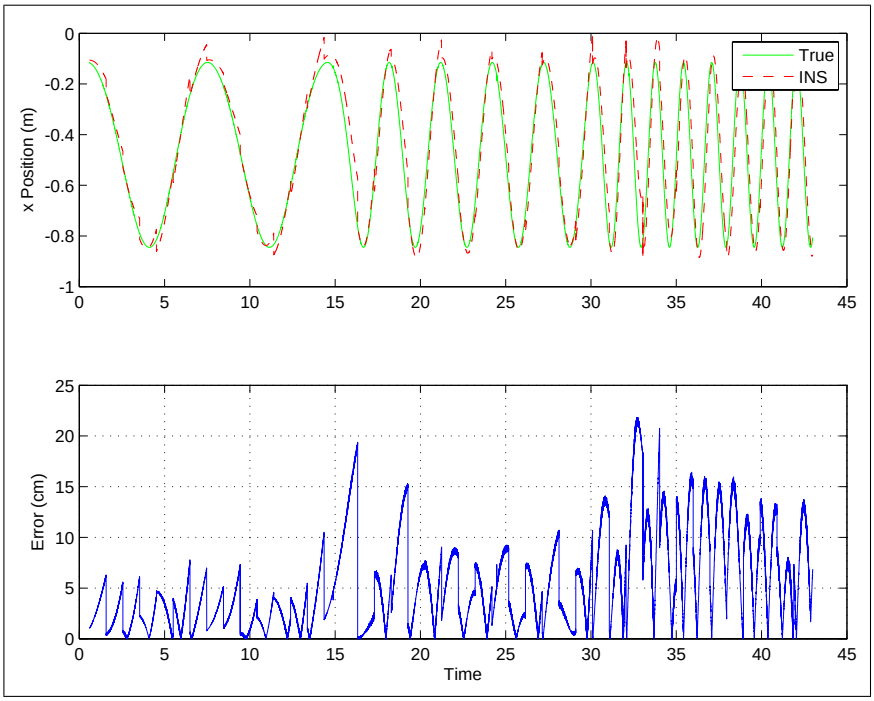


(c) x Velocity (m/s)

Figure 6.18: Magnetometer and optical encoder-aided INS velocity/position filter performance

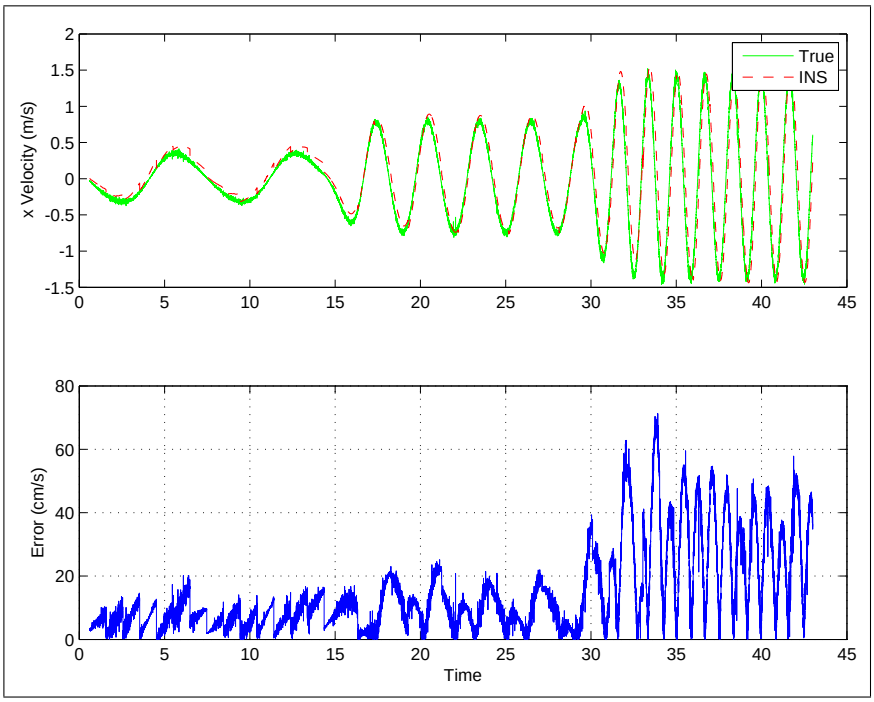


(a) Position (cm), rms error = 8.1 cm



(b) x-component of position (m)

Figure 6.19: Cricket-aided INS velocity/position filter performance. (rms values given for a platform angular velocity of -115 °/s.)



(c) x-component of velocity (m/s)

Figure 6.19: Cricket-aided INS velocity/position filter performance. (rms values given for a platform angular velocity of -115 °/s.)

The Cricket delays affect the heading measurements most adversely. Recall that there is a 100 ms delay between position estimates from individual Cricket units. When the devices are moving, the measurement from the first unit will be inconsistent when the second unit provides its measurement. Thus, the heading measurement error is dependent on the speed of the devices. Fortunately, the inaccuracies due to delay can be reduced by using data from the INS. The INS outputs may be used to estimate the first Cricket unit's position at the instant when the second unit provides its measurement.

Figure 6.20 illustrates the benefits of using the INS outputs to assist heading determination. The true heading is shown in solid blue, the raw Cricket heading is plotted in dashed red, and the INS-assisted heading is shown in dashed green. The data was obtained for a platform angular velocity of -115 °/s. Observe that the error drops from approximately 25° to 10° when the INS position estimates are utilized.

6.2.1.5 Summary

The INS experimental results are summarized in Table 6.7. The table compares the INS root mean square error for the different aiding sources. The data shows that two properly calibrated Cricket units are quite effective as an aiding source.

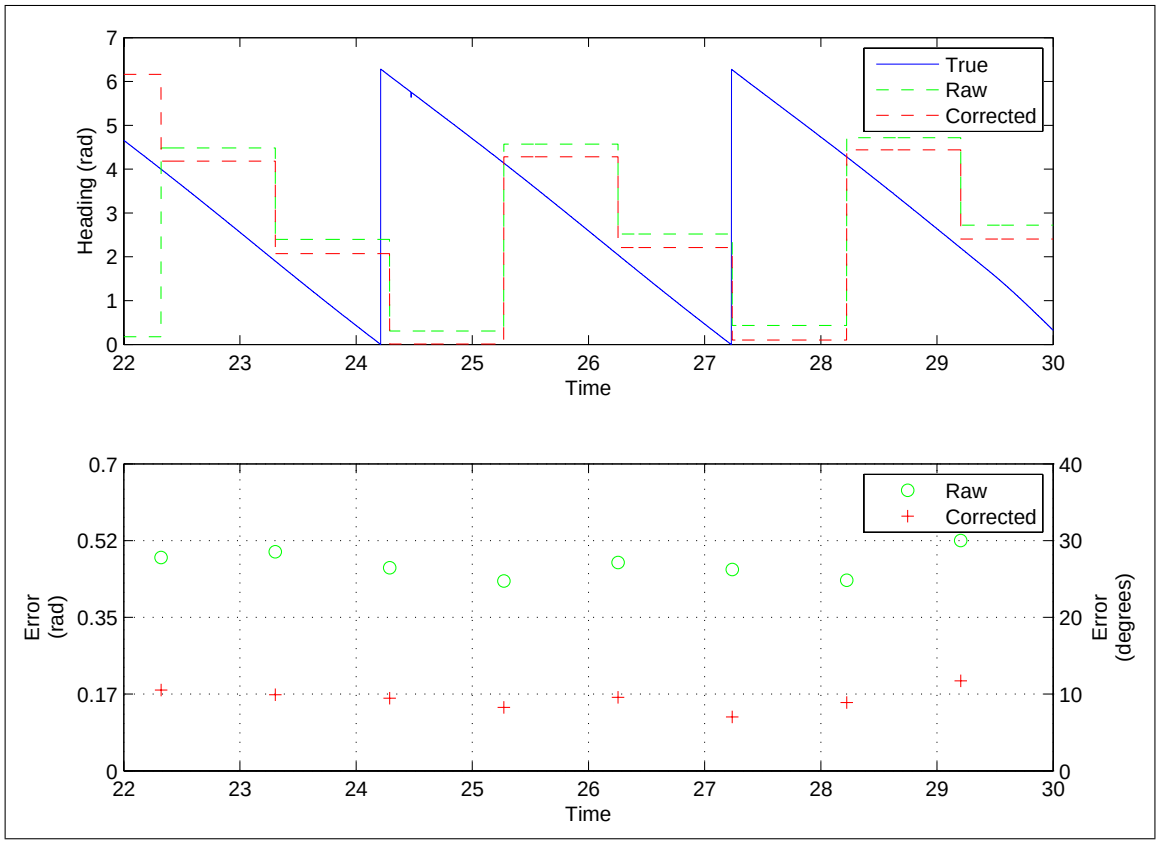


Figure 6.20: True heading reference compared with Cricket heading measurements at a platform angular velocity of -115 °/s. The INS-assisted Cricket heading measurements are approximately 15° more accurate.

Aiding Source & Platform Velocity	**Angular Velocity**	**Heading**	**Velocity**	**Position**
Magnetometer/ Optical Encoder (-166 °/s)	3.7 °/s	.5°	7.5 cm/s	3.4 cm
Cricket (-115 °/s)	4.1 °/s	9.8°	16.8 cm/s	8.1 cm

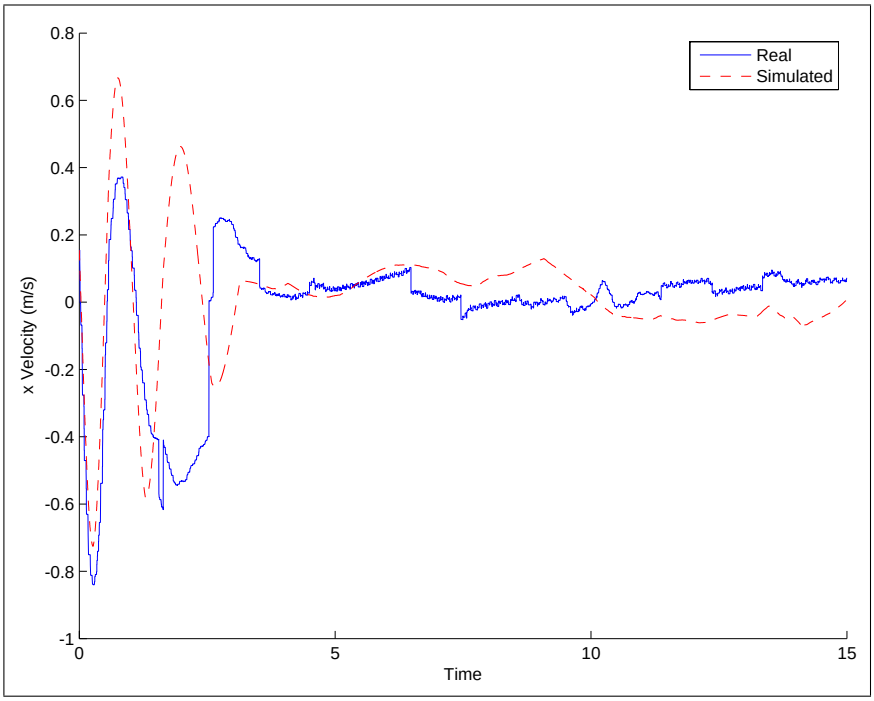
Table 6.7: Summary of INS errors (rms)

6.2.2 Hovercraft Autopilot Performance

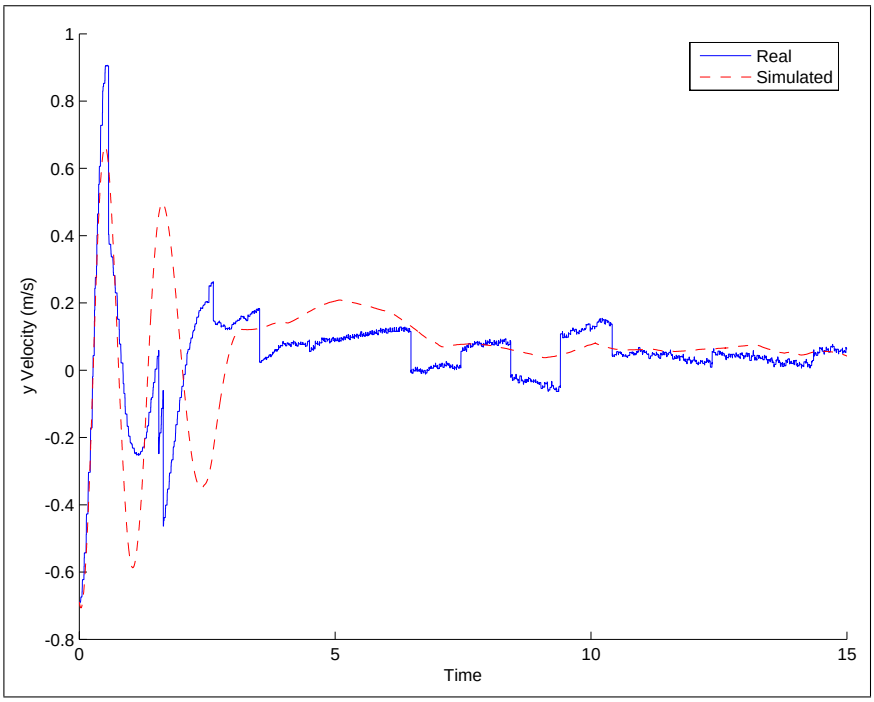
In this section, we present the results obtained by running the real autopilot system on the actual R/C hovercraft. For comparison with the simulated results, the initial conditions and parameters are the same. Thus, the simulated plots (shown in dashed red) are identical to those presented in section 6.1.1. In general, the agreement between the experimental and simulated results is quite good. Some of the plots, however, exhibit dynamical effects not captured by our model.

6.2.2.1 Zero Velocity Stabilization

Figures 6.21a - 6.21c depict the zero velocity stabilization results. The physical autopilot stabilizes the angular velocity faster than predicted by the simulation. This may be due to frictional effects unmodeled in the simulator. Observe that the actual longitudinal and lateral velocities agree nicely with the simulated results. Each of the figures exhibits similar steady state performance between the experimental and simulated results.

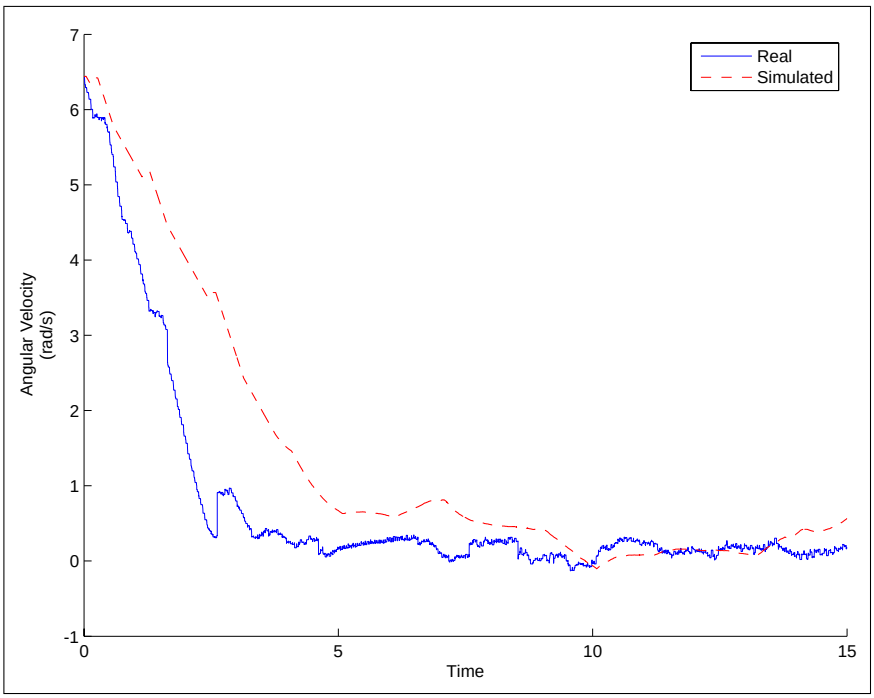


(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)

Figure 6.21: Experimental vs simulated results for zero velocity stabilization



(c) Angular velocity (Ω)

Figure 6.21: Experimental vs simulated results for zero velocity stabilization

6.2.2.2 Forward Velocity Stabilization

Figures 6.22a - 6.22c compare the experimental and simulated results for stabilizing a constant forward velocity of 1.1 m/s. The agreement between the plots is excellent. The experimental and simulated transient responses for the longitudinal velocity are practically identical. In addition, the steady state errors exhibit similar variance. As predicted by the simulation, the lateral and angular velocities fluctuate wildly about zero, but have small peak amplitudes.

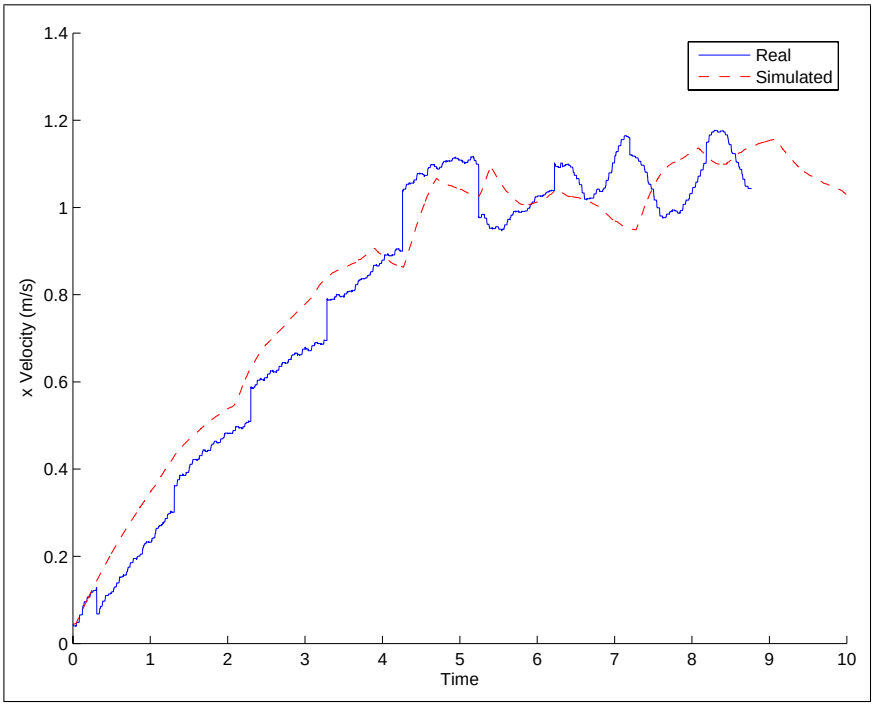
6.2.2.3 Reverse Velocity Stabilization

Figures 6.23a - 6.23c show the results for reverse velocity stabilization. Unfortunately, the plots are not quite as impressive as those produced for the forward velocity case. The desired longitudinal velocity was set to -.76 m/s.

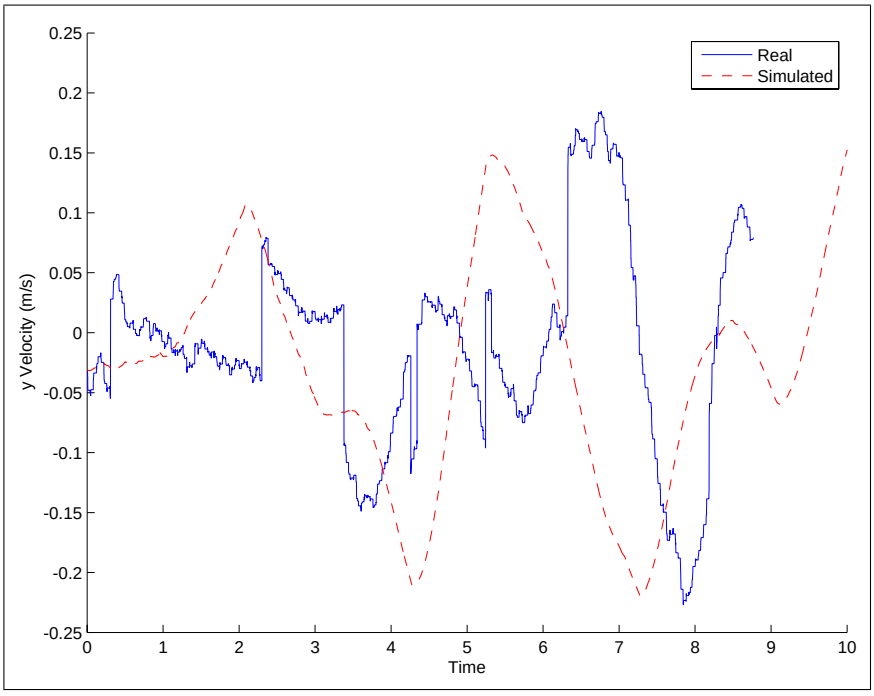
Figure 6.23a indicates the presence of nontrivial steady state error in the P_X state variable. Regardless of the selected reference velocity, the real autopilot was never able to fully eliminate the error. The source of the steady state error is not entirely clear, but may be due to unmodeled friction or thrust calibration errors. Note, however, that the steady state errors predicted by the simulation for the lateral and angular velocities agree well with the experimental results.

6.2.2.4 Constant Angular Velocity Stabilization

Steady state error is also present in the experimental results obtained for angular velocity stabilization. The results appear in Figures 6.24a - 6.24c for a

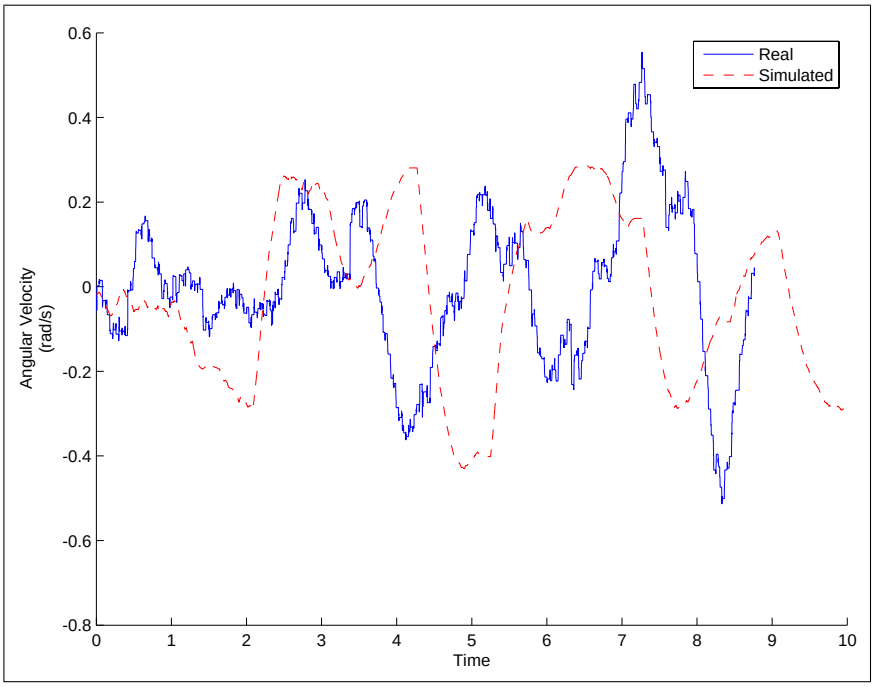


(a) Longitudinal velocity (V_X)



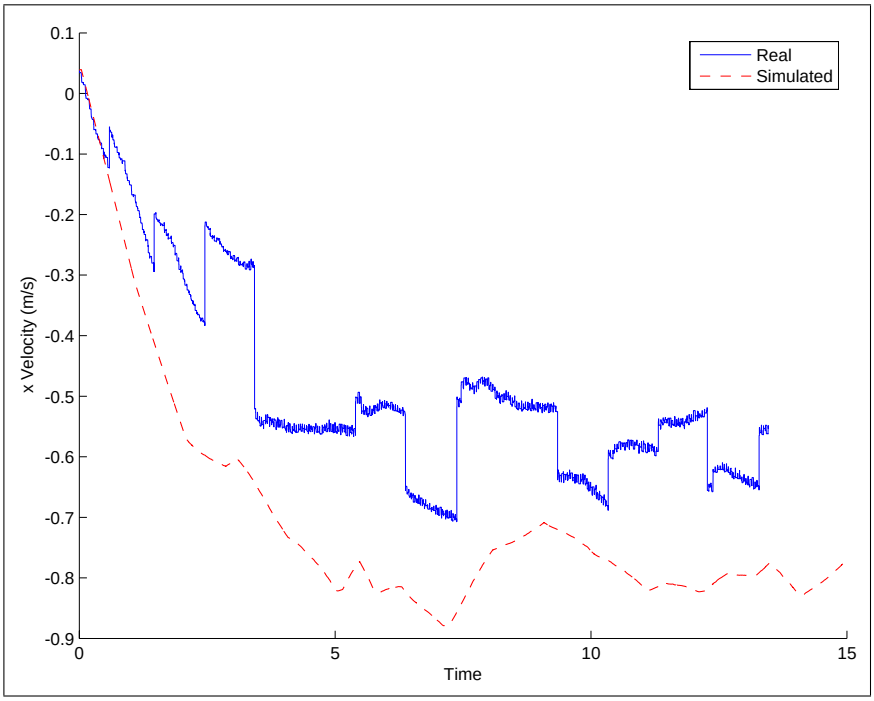
(b) Lateral velocity (V_Y)

Figure 6.22: Experimental vs simulated results for constant forward velocity stabilization ($\overline{V}_X = 1.1$ m/s)

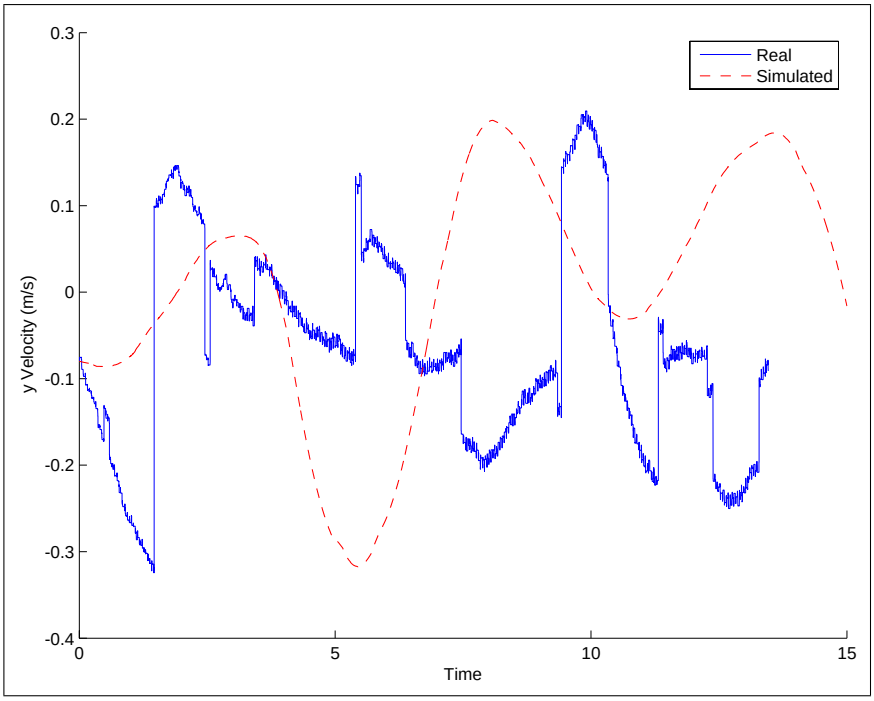


(c) Angular velocity (Ω)

Figure 6.22: Experimental vs simulated results for constant forward velocity stabilization

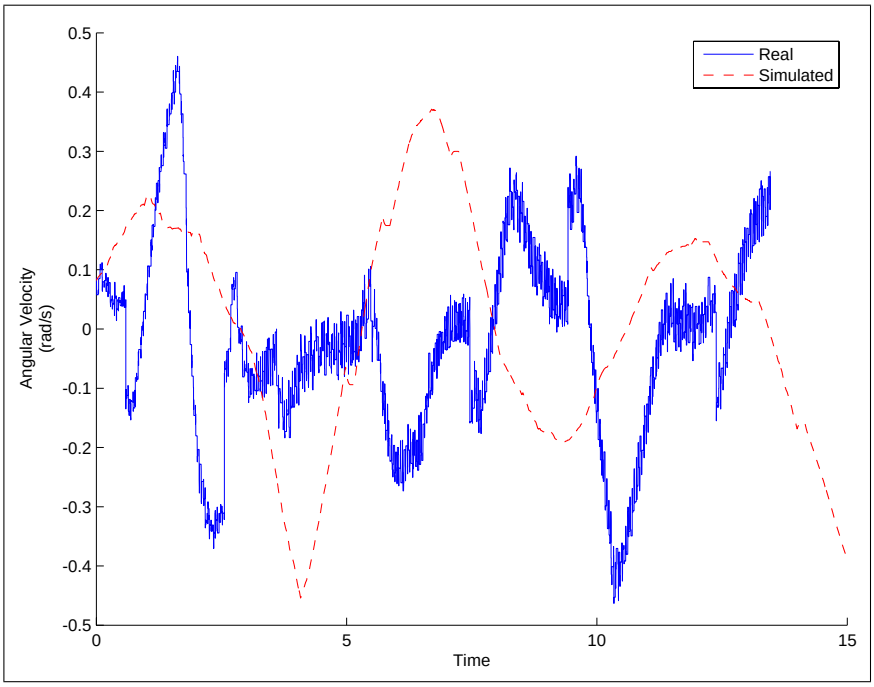


(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)

Figure 6.23: Experimental vs simulated results for constant reverse velocity stabilization



(c) Angular velocity (Ω)

Figure 6.23: Experimental vs simulated results for constant reverse velocity stabilization ($\overline{V}_X$ = -.76 m/s)

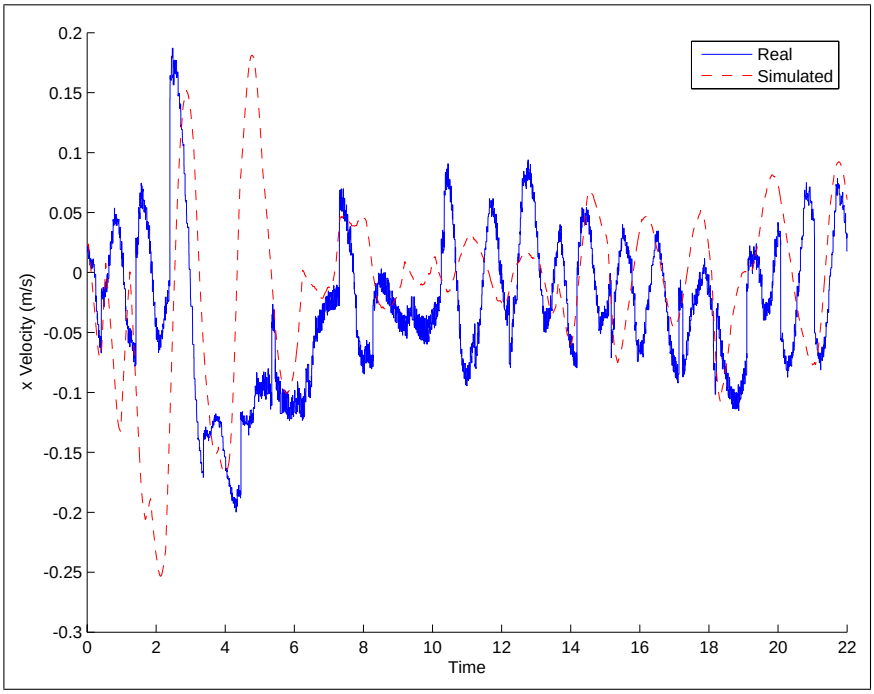
commanded reference angular velocity of 3 rad/s. While the simulated autopilot achieves the desired reference angular velocity, the actual autopilot falls short by about 40%. Figure 6.24c shows that the physical autopilot achieves and maintains a constant angular velocity of approximately 1.75 rad. We postulate that there are significant rotational friction effects not accounted for in our hovercraft model.

Despite the steady state error, the transient performance of the two autopilots is similar. The actual autopilot requires about 2 seconds longer to achieve its final angular velocity. Observe that the agreement between the simulated and experimental longitudinal velocity, V_X, is quite good. The experimental lateral velocity, V_Y, appears to have a small negative bias, however, while the simulated velocity does not.

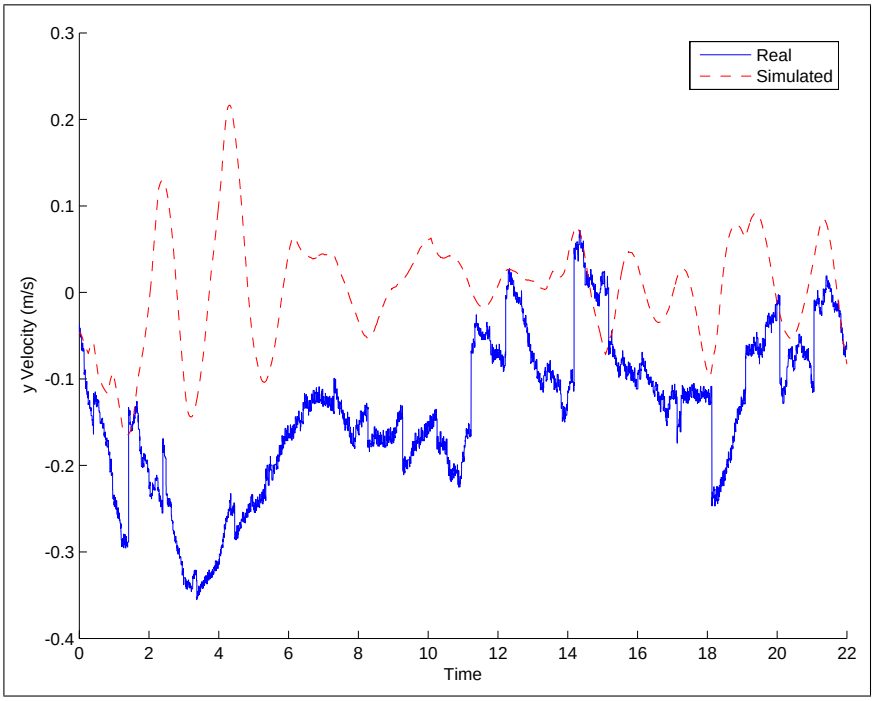
6.2.2.5 Heading Stabilization

The following set of plots present the experimental results for heading stabilization. For each trial, the hovercraft was initially at rest and pointing in the direction opposite to the desired 90° heading.

Figures 6.25a - 6.25d depict the experimental and simulated results for the bang-bang algorithm while Figures 6.26a - 6.26d compare the results for the proportional feedback algorithm. For both algorithms, the simulated heading transient response agrees nicely with the experimental results. The results for the remaining velocity variables also show good agreement. Observe that the experimental lat-

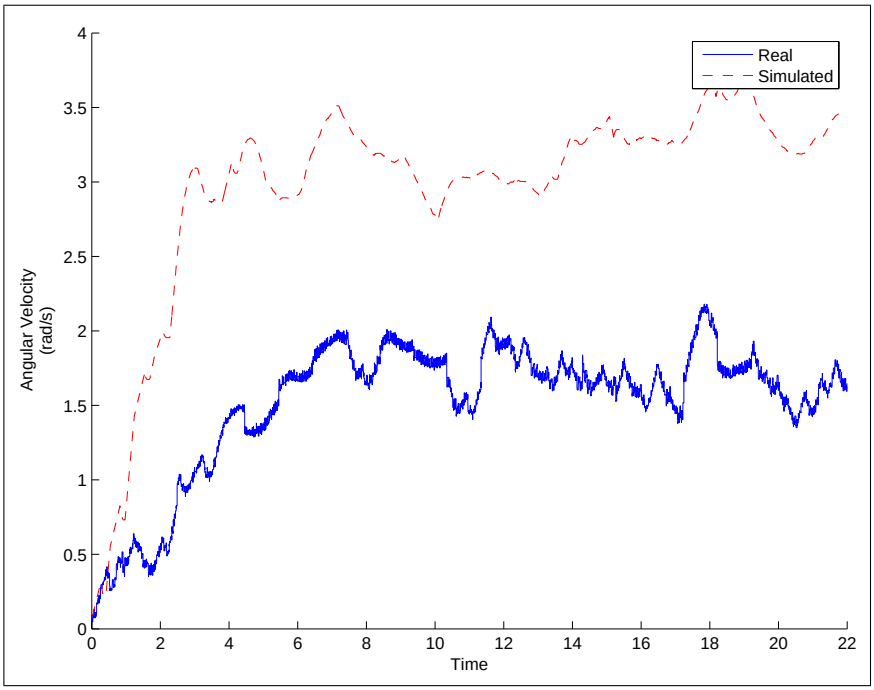


(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)

Figure 6.24: Experimental vs simulated results for constant angular velocity stabilization

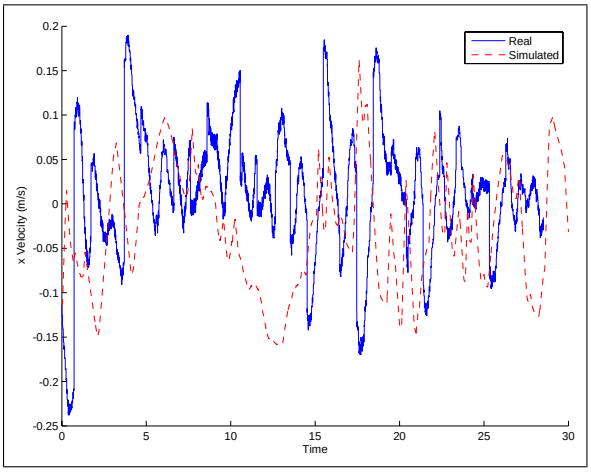


(c) Angular velocity (Ω)

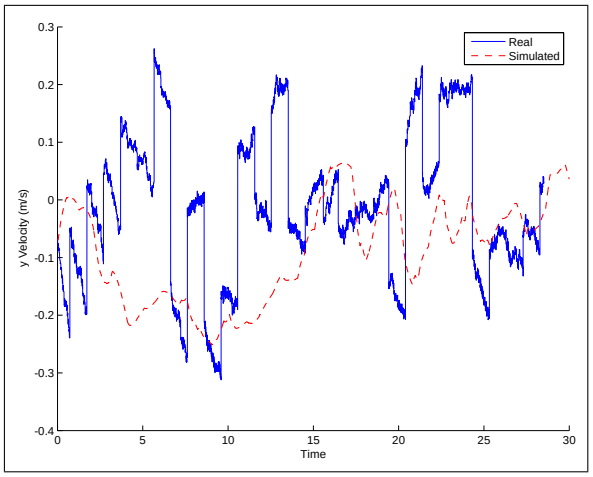
Figure 6.24: Experimental vs simulated results for constant angular velocity stabilization ($\overline{\Omega} = 3$ rad/s)

eral velocity, V_Y, for the bang-bang algorithm is more noisy than predicted by the simulator.

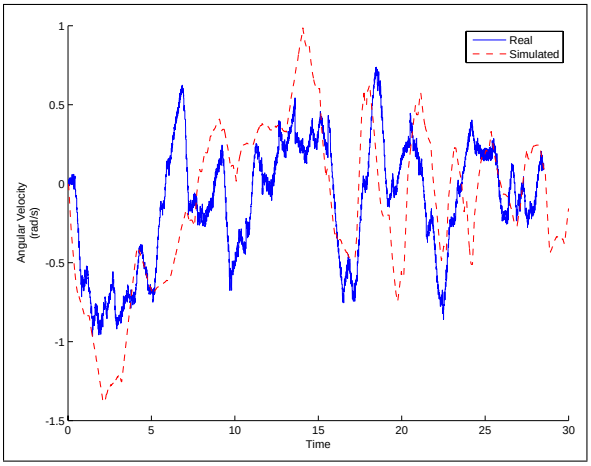
Finally, Figure 6.27 compares the experimental heading tracking performance for the two algorithms. As expected, the plots do not indicate any appreciable performance benefit from choosing one algorithm over the other.



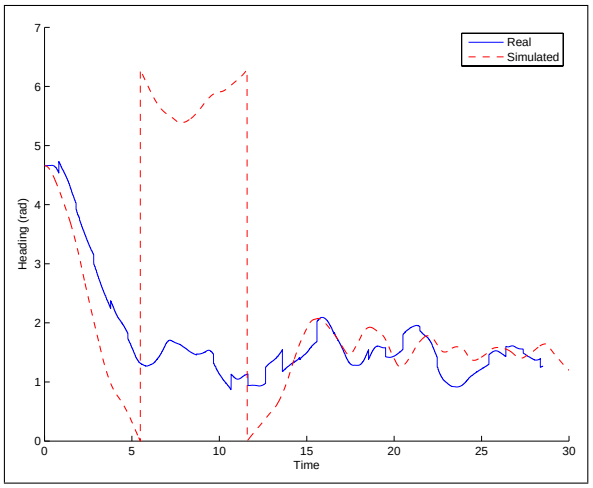
(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)

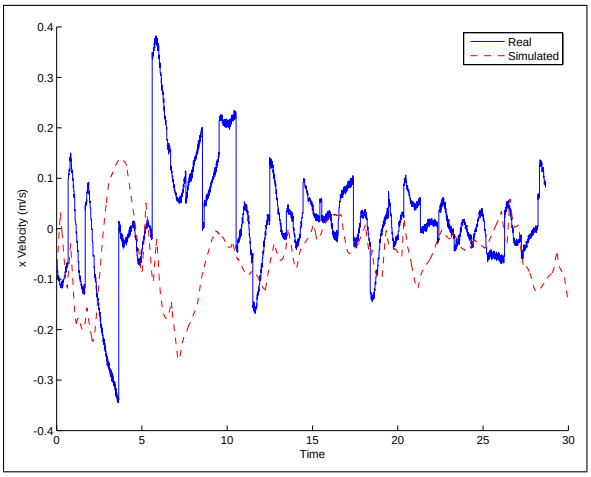


(c) Angular velocity (Ω)

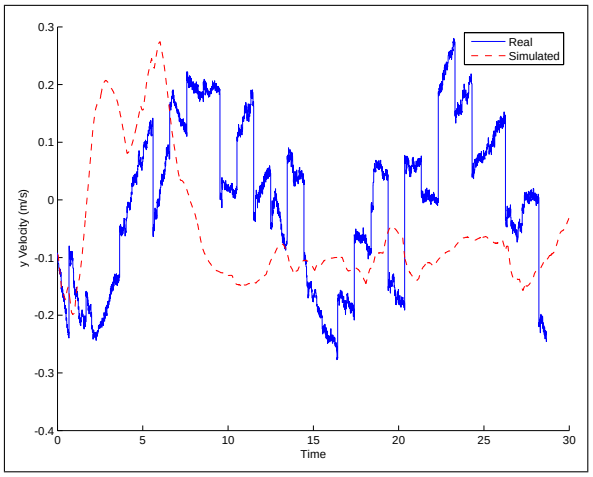


(d) Heading (θ)

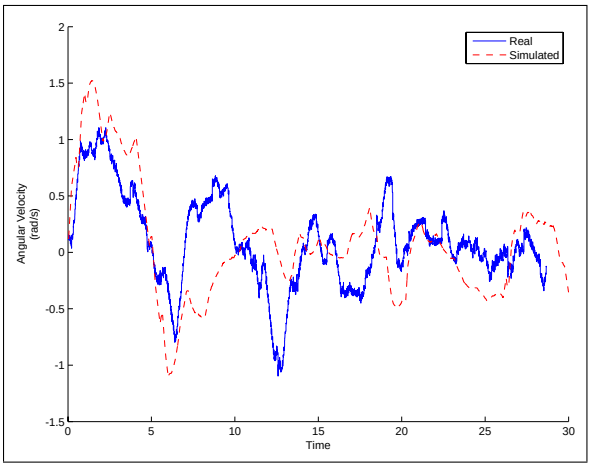
Figure 6.25: Experimental vs simulated results for heading stabilization (bang-bang algorithm)



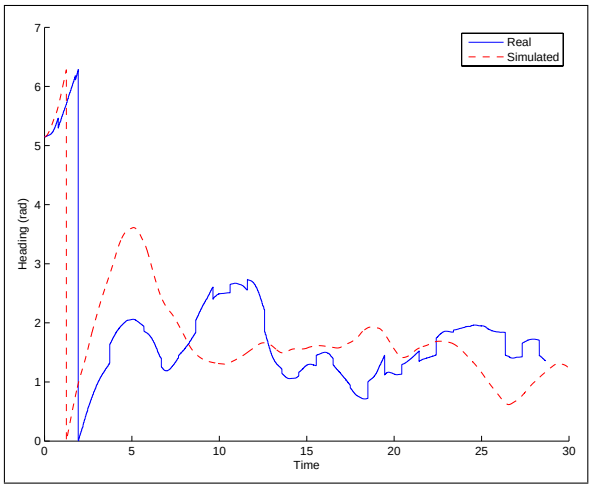
(a) Longitudinal velocity (V_X)



(b) Lateral velocity (V_Y)



(c) Angular velocity (Ω)



(d) Heading (θ)

Figure 6.26: Experimental vs simulated results for heading stabilization (proportional algorithm)

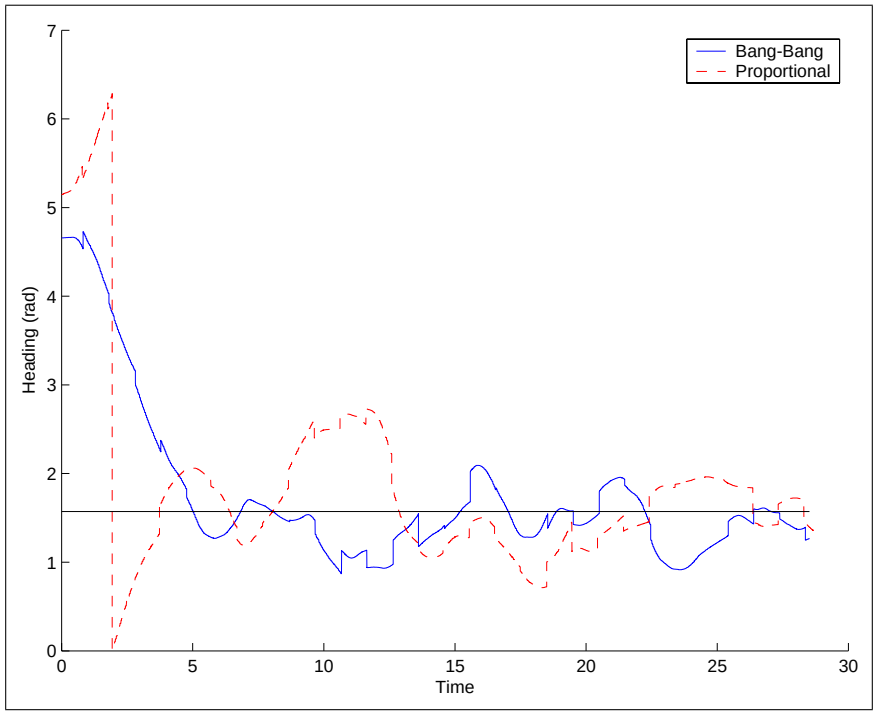


Figure 6.27: Experimental heading stabilization performance

Chapter 7

Conclusions and Future Research

Until recently, the development of a distributed control system for a model hovercraft would have been impractical. Necessary technologies such as inexpensive MEMs inertial sensors, lightweight, low power, and robust ad-hoc networking systems, and tools for rapid control system prototyping and evaluation were either prohibitively expensive or simply unavailable. Fortunately, new technologies are enabling the development of novel distributed control systems.

In this thesis, we have successfully demonstrated distributed automatic control of a real hovercraft using tools from the theory of nonlinear control. We derived a family of control laws for the reduced (ideal) hovercraft dynamics and experimentally demonstrated their ability to stabilize the (actual) hovercraft, even in the midst of large communication delays in the feedback loop. The control laws demonstrated served to stabilize zero velocity, constant forward/reverse velocity, constant angular velocity, and heading. In addition, we successfully developed and deployed a two-dimensional aided INS for measuring the vehicle state and demonstrated the effectiveness of the Cricket system as an indoor GPS replacement. We developed a high-fidelity simulation of the real autopilot system and showed the close agreement between simulated and experimental results. An invaluable tool, the simulator confirmed our suspicions that the limiting performance factor was navigation system

inaccuracy. Future control efforts will certainly address the INS and Cricket system for increased autopilot system performance.

The development of a fully autonomous hovercraft, capable of coordinating with other vehicles, is a natural goal. The control laws and methodologies presented in this thesis provide a solid foundation on which to augment the autopilot capabilities and strive toward full autonomy. The experimental results truly indicate the potential of the hovercraft autopilot and demonstrate the effectiveness of distributed control for complex nonlinear underactuated systems.

As we have seen, the dynamics of the underactuated hovercraft are deceivingly complex. In addition to the nonlinear convergence results presented here, averaging theory and geometric control are also important tools. For example, time-varying control laws have been developed to stabilize underactuated underwater vehicles in the event of an actuator failure [29].

In addition, we would like to develop a control law to track an arbitrary reference trajectory. One possible solution would be to combine the functionality of a path planner with a hybrid control strategy, making use of the control laws presented in this thesis for the reduced dynamics. An event-driven controller would be implemented to switch between the various continuous-time control laws as needed to minimize the tracking error. Such a control strategy could also be used to provide pilot-in-the-loop functionality. In this mode of operation, a human pilot would function as the path planner and specify desired velocity and turn rates via a joystick input device. Before implementing this hybrid control scheme, however, we would still need to derive a control law (for the ideal hovercraft reduced dynamics) to steer

the hovercraft while moving forward or in reverse. A time-varying perturbation signal superimposed on the force outputs of the $\overline{P}_X$ feedback law might provide the desired behavior.

Another option for trajectory tracking would be to implement a full six-dimensional control law to position the hovercraft at an arbitrary location and heading. We realize that this full dynamics control approach would be considerably more difficult than the hybrid control methodology mentioned previously.

In addition to refining the control and navigation laws, there are practical opportunities to improve the vehicle operation: e.g., enhancing the inertial navigation system, augmenting the actuation, lightening the vehicle, performing the processing onboard the vehicle, and providing additional sensing functionalities.

Regarding the INS, we have since discovered that better navigation performance can be achieved by using the raw ranges from the Cricket devices rather than the computed position estimates. This is significant because simulation shows that navigation error limits the attainable controller performance. An Extended Kalman Filter may be used to incorporate the available range estimates, as the range equations are nonlinear functions of the state variables. Alternatively, there has been recent success using nonlinear filtering techniques, such as particle filtering, to achieve navigation performance superior to Extended Kalman Filtering [30].

Using raw range measurements to aid the INS offers several performance advantages over the position-aided approach. First, all available range information is utilized. In contrast, recall that the vehicle's position can not be determined unambiguously when fewer than three ranges are available. As a result, important range

measurements are unnecessarily discarded in the position-aided approach. Second, there is significant time-correlation in the position errors over short intervals. This violates the basic white Gaussian noise hypothesis of Kalman Filtering. Third, using raw range measurements from two rigidly affixed Cricket units allows the filter to estimate heading more optimally than using raw position estimates from the devices. Since range information will be incorporated from multiple sensors, care must be taken to ensure proper lever arm compensation. The lever arm effect occurs whenever body-fixed sensors are offset from the origin of the INS reference frame. A clear explanation of the lever arm effect and its proper compensation in the INS equations is provided in [31].

Concerning vehicle design, there are opportunities to lighten the hovercraft and upgrade the actuation capabilities. Both of these modifications would increase the hovercraft's agility and lessen the burden on the controller. For example, the vehicle weight could be reduced through mechanical optimizations and electronics redesign. Creating a custom electronics board would save roughly one half pound in total vehicle weight. A significant upgrade to the thruster could be realized by using a variable-pitch propeller. These propellers spin at a constant RPM and use a small servo and cable to vary the blade pitch, thereby modulating the produced thrust. There are several advantages to designing a custom variable-pitch propeller for the hovercraft. Most importantly, a variable-pitch propeller would obviate the need for a reversing speed controller (eliminating the problematic reverse delay) and would prolong the life of both the speed controller and motor. The improvement in actuation bandwidth would greatly improve the autopilot transient response.

Finally, the local processing capabilities of the autopilot could be upgraded to run the control laws and navigation filters onboard the vehicle. Eliminating the network latencies in the control loop would improve the autopilot performance. Instead of routing sensor data and actuation commands, Bluetooth or a similar ad-hoc networking technology would provide reduced bandwidth peer-to-peer communications for multi-vehicle coordination in swarms. The distributed control of a swarm of autonomous hovercraft is highly ambitious, but the results presented in this thesis and the experimental experience gained will help to enable future research efforts in this direction.

Appendix A

Microcontroller Schematics

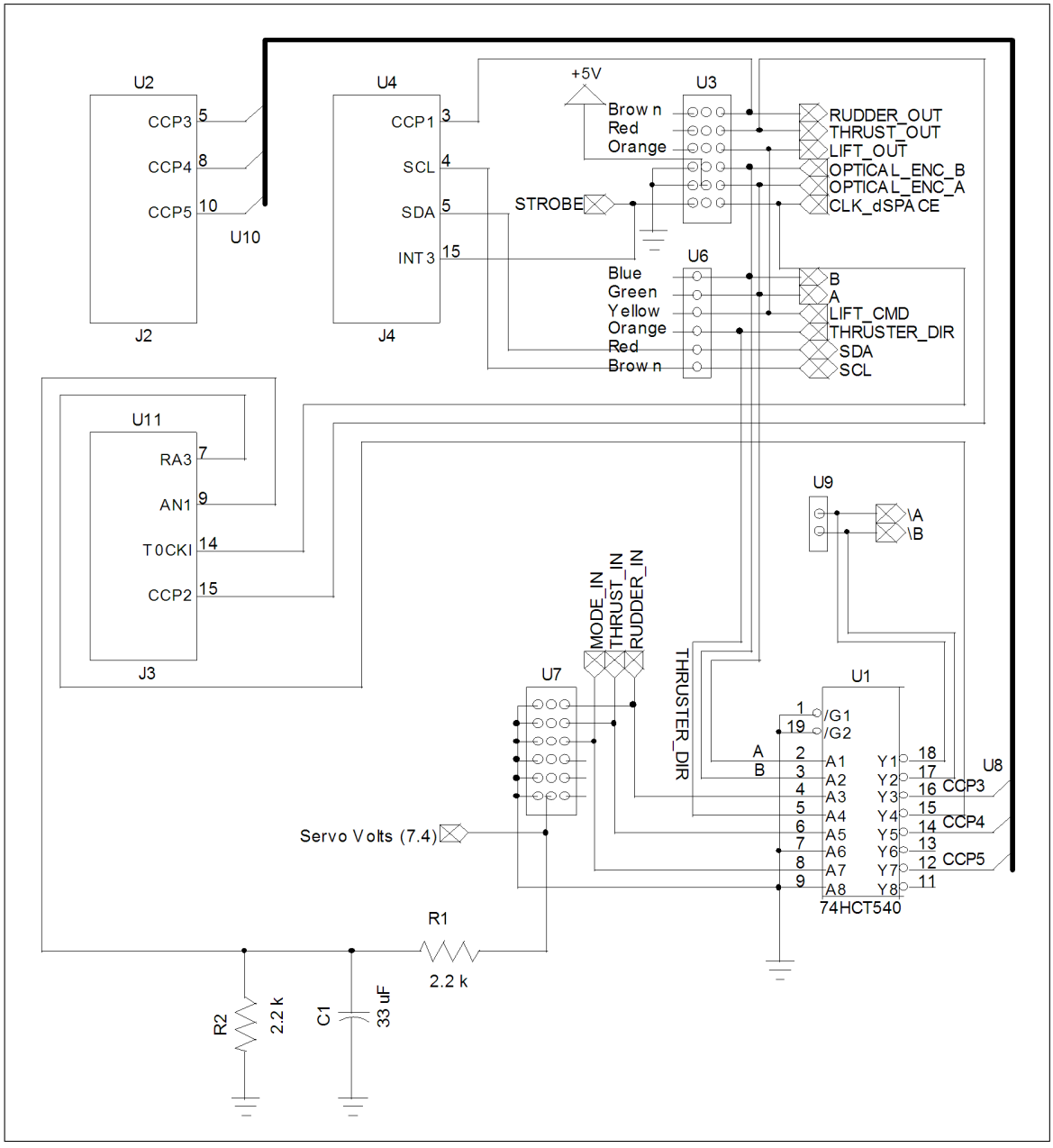


Figure A.1: Master microcontroller schematic

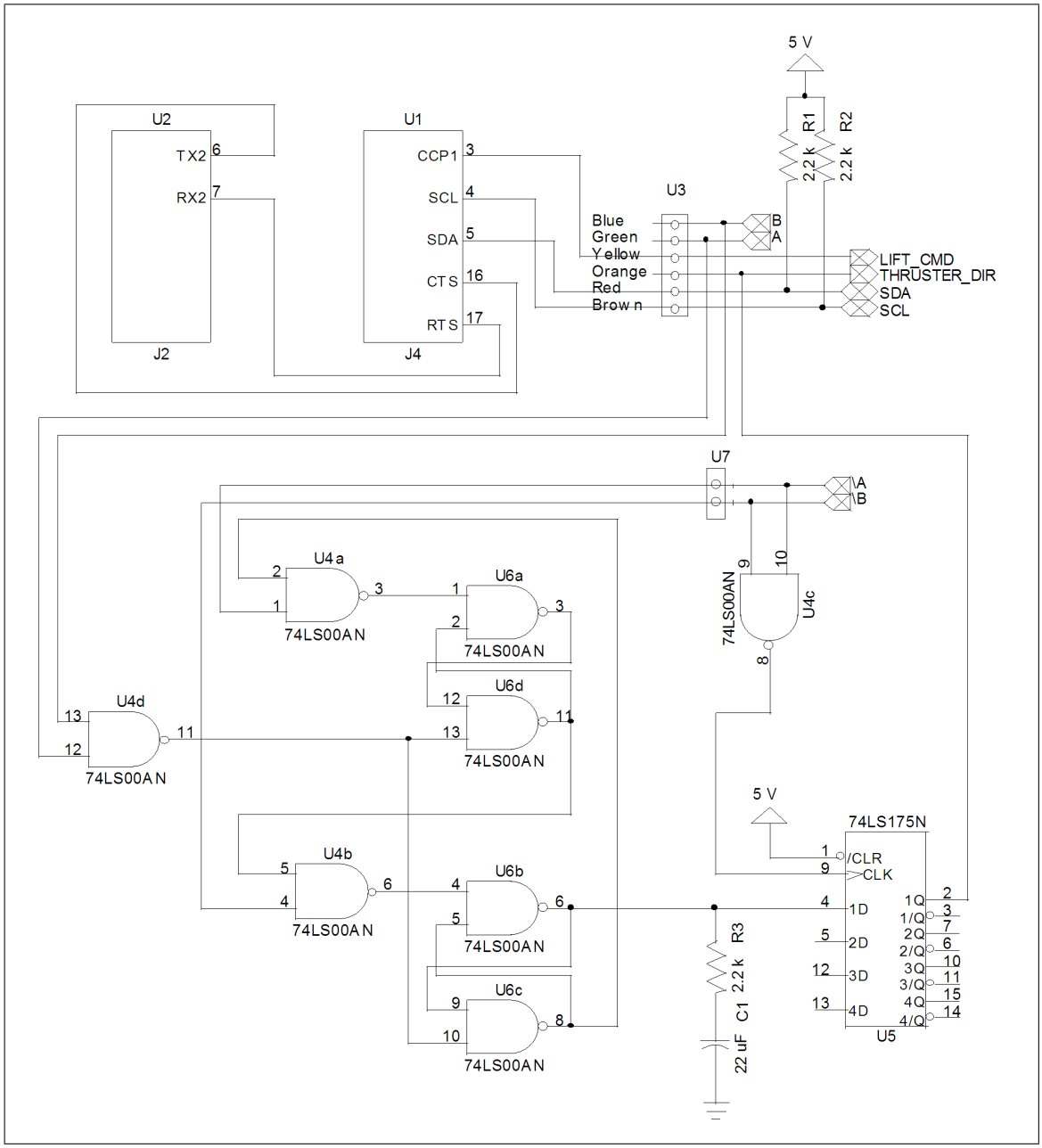


Figure A.2: Slave microcontroller schematic

Appendix B

Pilot Console

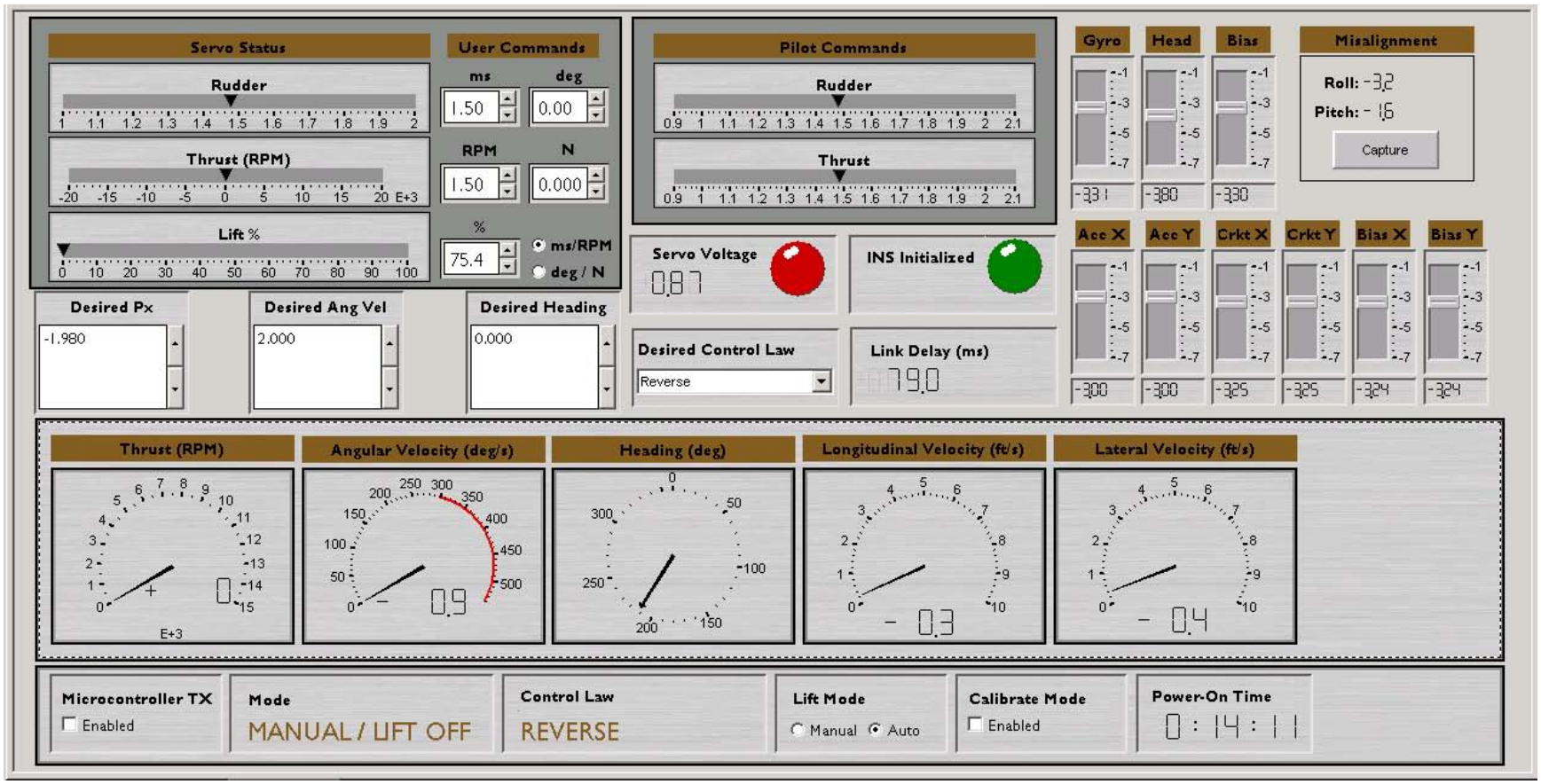


Figure B.1: Hovercraft pilot console

BIBLIOGRAPHY

[1] Robert L. Trillo. *Marine Hovercraft Technology*. L. Hill, London, 1971.

[2] I. Fantoni, R. Lozano, F. Mazenc, and K. Pettersen. Stabilization of a nonlinear underactuated hovercraft. *International Journal of Robust and Nonlinear Control*, 10(8):645–654, 2000.

[3] K. Y. Pettersen and O. Egeland. Exponential stabilization of an underactuated surface vessel. In *Proceedings of the 35th IEEE Conference on Decision and Control*, pages 967–972, New York, 1996. IEEE.

[4] C. I. Byrnes and A. Isidori. On the attitude stabilization of rigid spacecraft. *Automatica*, 27(1):87–95, 1991.

[5] António Pedro Aguiar, Lars Cremean, and João Pedro Hespanha. Position tracking for a nonlinear underactuated hovercraft: Controller design and experimental results. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 3858–3863, New York, 2003. IEEE.

[6] Hiroaki Seguchi and Toshiyuki Ohtsuka. Nonlinear receding horizon control of an RC hovercraft. In *Proceedings of the 2002 IEEE International Conference on Control Applications*, pages 1076–1081, New York, 2002. IEEE.

[7] K.Y. Petersen and H. Nijmeijer. Tracking control of an underactuated surface vessel. In *Proceedings of the 37th IEEE Conference on Decision and Control*, pages 4561–4566, New York, 1998. IEEE.

[8] R. W. Brockett. Asymptotic stability and feedback stabilization. In R. S. Millman and H. J. Sussmann, editors, *Differential Geometric Control Theory*, pages 181–191, Boston, 1983. Birkhäuser.

[9] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 3rd edition, 2001.

[10] Vikram Manikonda. *Control and Stabilization of a Class of Nonlinear Systems with Symmetry*. PhD thesis, University of Maryland, 1998.

[11] Jay A. Farrell and Matthew Barth. *The Global Positioning System & Inertial Navigation*. McGraw-Hill, 1999.

[12] Kenneth R. Britting. *Inertial Navigation Systems Analysis*. Wiley-Interscience, New York, 1971.

[13] D. H. Titterton and J. L. Weston. *Strapdown Inertial Navigation Technology*. IEE radar, sonar, navigation, and avionics series, 5. Peter Peregrinus Ltd., 1997.

[14] Daniel Choukroun. *Novel Methods for Attitude Determination Using Vector Observations*. PhD thesis, Israel Institute of Technology, 2003.

[15] G. Wahba. A least squares estimate of satellite attitude. *SIAM Review*, 7(3):409, 1965.

[16] G. M. Lerner. Three-axis attitude determination. In J.R. Wertz, editor, *Spacecraft Attitude Determination and Control*, pages 420–428. D. Reidel, Dordrecht, 1978.

[17] J. D. Powell Demoz Gebre-Egziabher, Gabriel H. Elkaim and Bradford W. Parkinson. A gyro-free quaternion-based attitude determination system suitable for implementation using low cost sensors. In *Proceedings of the IEEE Position, Location, and Navigation Symposium*, pages 185–192, New York, 2000. IEEE.

[18] Cricket v2 user manual. http://cricket.csail.mit.edu/v2man.pdf.

[19] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket location-support system. In *Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking*, Boston, August 2000.

[20] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, 1979.

[21] Wilson J. Rugh. *Linear System Theory*. Information and System Sciences. Prentice-Hall, 2nd edition, 1996.

[22] http://www.ngdc.noaa.gov/seg/geomag/jsp/IGRF.jsp.

[23] J.R. Amyot, editor. *Hovercraft Technology, Economics, and Applications*, volume 11 of *Studies in Mechanical Engineering*. Elsevier, New York, 1989.

[24] Ian Cross and Coleman O'Flaherty. *Introduction to Hovercraft and Hoverports*. Pitman, London, 1975.

[25] Bo Bernhardsson, Johan Eker, and Joakim Persson. Bluetooth in control. In *Handbook of Networked and Embedded Control Systems*, pages 699–720. Birkhäuser, 2005.

[26] http://www.math.uiowa.edu/∼dstewart/meschach.

[27] Owen Cramer. The variation of the specific heat ratio and the speed of sound in air with temperature, pressure, humidity, and CO_2 concentration. *The Journal of the Acoustical Society of America*, 93(5):2510–2516, 1993.

[28] S. Bancroft. An algebraic solution of the GPS equations. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21(7):56–59, January 1985.

[29] N. E. Leonard. *Averaging and motion control of systems on Lie groups*. PhD thesis, University of Maryland, 1994.

[30] B. Boberg and S.-L. Wirkander. Integrating GPS and INS: comparing the Kalman estimator and particle estimator. In *7th International Conference on Control, Automation, Robotics and Vision*, pages 484–490, Boston, 2002.

[31] Jaewon Seo, Hyung Keun Lee, Jang Gyu Lee, and Chan Gook Park. Lever arm compensation for GPS/INS/Odometer integrated system. *International Journal of Control, Automation, and Systems*, 4(2):247–254, April 2006.